Barbara Catania
Mirjana Ivanović
Bernhard Thalheim (Eds.)

LNCS 6295

Advances in Databases and Information Systems

14th East European Conference, ADBIS 2010
Novi Sad, Serbia, September 2010
Proceedings

Springer

Barbara Catania Mirjana Ivanović
Bernhard Thalheim (Eds.)

Advances in Databases and Information Systems

14th East European Conference, ADBIS 2010
Novi Sad, Serbia, September 20-24, 2010
Proceedings

Springer

Volume Editors

Barbara Catania
University of Genova
Department of Computer and Information Science
Via Dodecaneso, 35, 16146 Genova, Italy
E-mail: catania@disi.unige.it

Mirjana Ivanović
University of Novi Sad, Faculty of Science
Department of Mathematics and Informatics
Trg Dositeja Obradovica 4, 21000 Novi Sad, Serbia
E-mail: mira@dmi.uns.ac.rs

Bernhard Thalheim
Christian-Albrechts-University of Kiel
Institute of Computer Science and Applied Mathematics
Olshausenstr. 40, 24098 Kiel, Germany
E-mail: thalheim@is.informatik.uni-kiel.de

Library of Congress Control Number: Applied for

CR Subject Classification (1998): H.2, H.4, H.3, C.2, J.1, H.2.8

LNCS Sublibrary: SL 3 – Information Systems and Application, incl. Internet/Web and HCI

ISSN	0302-9743
ISBN-10	3-642-15575-8 Springer Berlin Heidelberg New York
ISBN-13	978-3-642-15575-8 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper 06/3180

Preface

This volume contains the best papers presented at the 14th East-European Conference on Advances in Databases and Information Systems (ADBIS 2010), held during September 20-24, 2010, in Novi Sad, Serbia.

ADBIS 2010 continued the ADBIS series held in St. Petersburg (1997), Poznan (1998), Maribor (1999), Prague (2000), Vilnius (2001), Bratislava (2002), Dresden (2003), Budapest (2004), Tallinn (2005), Thessaloniki (2006), Varna (2007), Pori (2008), and Riga (2009). The main objective of the ADBIS series of conferences is to provide a forum for the dissemination of research accomplishments and to promote interaction and collaboration between the database and information systems research communities from Central and East European countries and the rest of the world. The ADBIS conferences provide an international platform for the presentation of research on database theory, development of advanced DBMS technologies, and their advanced applications.

ADBIS 2010 spans a wide area of interests, covering all major aspects related to theory and applications of database technology and information systems. Two different submission lines were considered for ADBIS 2010, one within the classic track and another one within a special track organisation. ADBIS comprised five tracks:

1. Conceptual Modeling in Systems Engineering (CMSE)
2. Data Mining and Information Extraction (DMIE)
3. Business Processes in E-Commerce Systems (e-commerce)
4. Personal Identifiable Information: Privacy, Ethics, and Security (PIIPES)
5. Warehousing and OLAPing Complex, Spatial and Spatio-Temporal Data (WOCD)

In total, the conference attracted 165 paper submissions from Algeria, Armenia, Australia, Austria, Bosnia and Herzegovina, Brazil, Bulgaria, Canada, Chile, China, Czech Republic, Estonia, France, Germany, Greece, Hungary, India, Ireland, Italy, Jordan, Lithuania, FYR Macedonia, Malaysia, New Zealand, Nigeria, Poland, Portugal, Republic of Korea, Romania, Russia, Serbia, Slovakia, Spain, Sweden, Tunisia, Turkey, Ukraine, and USA. In a rigorous reviewing process, 36 papers were selected for inclusion in these proceedings as long contributions. Moreover, we selected 14 papers as short contributions. All papers were evaluated by at least three reviewers. The selected papers span a wide spectrum of topics in the database and information systems field, including database theory, database management system architectures, design methods, data mining and data warehousing, spatial, temporal, and moving objects, database applications.

We would like to thank everyone who contributed to the success of ADBIS 2010. We thank the authors, who submitted papers to the conference. A special thanks to the Track Co-chairs, to the main conference and track Program

Committee members as well as to the external reviewers, for their support in in evaluating the papers submitted to ADBIS 2010, ensuring the quality of the scientific program. We thank our colleagues and the students of our universities for their help in the workshop organization. Thanks to all members of the local organizing team in Novi Sad, for giving their time and expertise to ensure the success of the conference. Finally, we thank Springer for publishing the proceedings containing invited and research papers in the LNCS series. The Program Committee work relied on MuCoMS, which proved to be a very convenient tool for this kind of work, and we are grateful to the MuCoMS development team and to its leader, Markus Kirchberg, who created and maintain it. Last but not least we thank the Steering Committee and, in particular, its Chair, Leonid Kalinichenko, for their help and guidance. Last, but not least, we thank the participants of ADBIS'2010 for having made our work useful.

September 2010

Barbara Catania
Mirjana Ivanović
Bernhard Thalheim

Conference Organization

General Chair

Mirjana Ivanović	University of Novi Sad, Serbia

Program Co-chairs

Barbara Catania	University of Genova, Italy
Bernhard Thalheim	Christian Albrechts University of Kiel, Germany

Program Committee

Costin Badica (Romania)
Guntis Barzdins (Latvia)
Alberto Belussi (Italy)
Andras Benczur (Hungary)
Maria Bielikova (Slovakia)
Omar Boucelma (France)
Stephane Bressan (Singapore)
Zoran Budimac (Serbia)
Dumitru Burdescu (Romania)
Albertas Caplinskas (Lithuania)
Boris Chidlovskii (France)
Alfredo Cuzzocrea (Italy)
Johann Eder (Austria)
Pedro Furtado (Portugal)
Matteo Golfarelli (Italy)
Giovanna Guerrini (Italy)
Hele-Mai Haav (Estonia)
Leonid Kalinichenko (Russia)
Damir Kalpić (Croatia)
Ahto Kalja (Estonia)
Mehmed Kantardžić (USA)
Panagiotis Karras (Singapore)
Sergei Kuznetsov (Russia)
Ivan Luković (Serbia)
Federica Mandreoli (Italy)
Rainer Manthey (Germany)
Manuk Manukyan (Armenia)
Joris Mihaeli (Istrael)
Paolo Missier (UK)
Tadeusz Morzy (Poland)
Alexandros Nanopoulos (Germany)
Pavol Navrat (Slovakia)
Mykola Nikitchenko (Ukraine)
Boris Novikov (Russia)
Gordana Pavlović-Lazetić (Serbia)
Jaroslav Pokorny (Czech Republic)
Stefano Rizzi (Italy)
Paolo Rosso (Spain)
Ismael Sanz (Spain)
Vaclav Snasel (Czech Republic)
Predrag Stanišić (Montenegro)
Bela Stantic (Australia)
Dragan Stojanović (Serbia)
Manolis Terrovitis (Greece)
Stefan Trausan Matu (Romania)
Athena Vakali (Greece)
Olegas Vasilecas (Lithuania)
Panos Vassiliadis (Greece)
Goran Velinov (FYR Macedonia)
Tatjana Welzer (Slovenia)
Marek Wojciechowski (Poland)
Limsoon Wong (Singapore)

Tracks Co-chairs

Sabah Al-Fedaghi	PIIPES	Kuwait University, Kuwait
Mikaël Ates	PIIPES	France Entr'Ouvert, Paris, France
Alfredo Cuzzocrea	WOCD	University of Calabria, Italy
Ajantha Dahanayake	CMSE	Georgia College and State University, USA
Andreas Speck	e-Commerce	Christian-Albrechts-Universität zu Kiel, Germany
Milan Zorman	Data mining	Maribor University, Slovenia

Track Program Committee Members

Alberto Abello
Fabrizio Angiulli
Joshep Barjis
Ladjel Bellatreche
Francesco Buccafurri
Richard Chbeir
Frederic Cuppens
Ernesto Damiani
Antonios Deligiannakis
Curtis Dyreson
Todd Eavis
Jacques Fayolle
Filippo Furfaro
Douglas Goings
Chen Gongliang
Jos van Hillegersberg
Hannu Jaakkola
Philippe Jaillon
Yoshio Kakizaki
Peter Kokol
Anne Laurent
Maryline Laurent
Gianluca Lax
Jens Lechtenborger
Jason Li
Loick Loth
Pat Martin
Pierre Parrend
Marc Pasquet
Torben Bach Pedersen
Sudha Ram
Mirek Riedewald
Michel Riguidel
Matti Rossi
Andreas Rusnjak
Marcel Schulz
Alkis Simitsis
Henk Sol
Arne Solvberg
Zoran Stojanovic
David Taniar
Panos Vassiliadis
Alexander Verbraeck
Laurent Vercouter
Wei Wang
Richard Welke
Robert Wrembel
Howard Woodard
Ma Ying Hua
Yoo Jang-Hee
Karine Zeitouni
Bin Zhou
Esteban Zimanyi

Additional Reviewers

Irina Astrova
Eftychia Baikoussi
Michal Barla
Andreas Behrend
Carlo Combi
Claus Dabringer
Bogdan Franczyk
Marko Holbl
Julius Koepke
Christian Koncilia
Stefano Lodi
Saša Malkov
Riccardo Martoglia
Yosi Mass
Sara Migliorini
Margus Nael
Barbara Oliboni
Marco Patella
Pavle Mogin
Wilma Penzo
David Eduardo
Janari Pold
Margus Pold
Vlad Posea
Elke Pulvermüller
Traian Rebedea
Sonja Ristić
Tarmo Robal
Simona Sassatelli
Marian Simko
Irena Spasić
Wee Hyong Tok

Organizing Committee

Zoran Putnik	Organizing General Chair	University of Novi Sad, Serbia
Dejan Mitrović	Secretary	University of Novi Sad, Serbia
Markus Kirchberg	Paper Reviewing and Submission System	A*STAR, Singapore

Members (all from University of Novi Sad)

Miroslav Vesković
Ivan Luković
Miloš Radovanović
Ivan Pribela
Saša Tošić
Vladimir Kurbalija
Jovana Vidaković
Živana Komlenov
Gordana Rakić
Doni Pracner
Milan Ćeliković

Organizing Institutions

Faculty of Sciences, University of Novi Sad, Serbia
Faculty of Technical Sciences, University of Novi Sad, Serbia

ADBIS Steering Committee

Chairman: Leonid Kalinichenko, Russian Academy of Science, Russia

Paolo Atzeni (Italy)
Andras Benczur (Hungary)
Albertas Caplinskas (Lithuania)
Johann Eder (Austria)
Marite Kirikova (Latvia)
Hele-Mai Haav (Estonia)
Mirjana Ivanović (Serbia)
Hannu Jaakkola (Finland)
Mikhail Kogalovsky (Russia)
Yannis Manolopoulos (Greece)
Rainer Manthey (Germany)
Manuk Manukyan (Armenia)
Joris Mihaeli (Israel)
Tadeusz Morzy (Poland)
Pavol Navrat (Slovakia)
Boris Novikov (Russia)
Mykola Nikitchenko (Ukraine)
Jaroslav Pokorny (Czech Republic)
Boris Rachev (Bulgaria)
Bernhard Thalheim (Germany)
Gottfried Vossen (Germany)
Tatjana Welzer (Slovenia)
Viacheslav Wolfengagen (Russia)
Ester Zumpano (Italy)

Sponsoring Institutions of the ADBIS 2010 Conference

ADBIS 2010 Conference was partially financially supported by:

Ministry of Science and Technological Development, Republic of Serbia
Provincial Secretariat for Science and Technological Development
Autonomous Province of Vojvodina, Republic of Serbia

Table of Contents

Reasoning with Imperfect Context and Preference Information in Multi-context Systems

G. Antoniou, A. Bikakis, and C. Papatheodorou

Institute of Computer Science, FO.R.T.H.
Vassilika Vouton, P.O. Box 1385, GR 71110
Heraklion, Greece
{Antoniou,bikakis,cpapath}@ics.forth.gr

Abstract. Multi-Context Systems (MCS) are logical formalizations of distributed context theories connected through a set of mapping rules, which enable information flow between different contexts. Recent studies have proposed adding non-monotonic features to MCS to handle problems such as incomplete, uncertain or ambiguous context information. In previous work, we proposed a non-monotonic extension to MCS and an argument-based reasoning model that enable handling cases of imperfect context information based on defeasible reasoning. To deal with ambiguities that may arise from the interaction of context theories through mappings, we used a preference relation, which is represented as a total ordering on the system contexts. Here, we extend this approach to additionally deal with incomplete preference information. To enable this, we replace total preference ordering with partial ordering, and modify our argumentation framework and the distributed algorithms that we previously proposed to meet the new requirements.

1 Introduction

A Multi-Context System consists of a set of *contexts* and a set of inference rules (known as *mapping* rules) that enable information flow between different contexts. A context can be thought of as a logical theory (a set of axioms and inference rules) that models local knowledge. Different contexts are expected to use different languages, and although each context may be locally consistent, global consistency cannot be required or guaranteed.

The notions of *context* and *contextual reasoning* were first introduced in AI by McCarthy [1], as an approach for the problem of *generality*. In the same paper, he argued that the combination of non-monotonic reasoning and contextual reasoning would constitute an adequate solution to this problem. Since then, two main formalizations have been proposed to formalize context: the Propositional Logic of Context, *PLC* [2,3], and the Multi-Context Systems introduced in [4], which later became associated with the Local Model Semantics proposed in [5]. Multi-Context Systems have been argued to be most adequate with respect to the three properties of contextual reasoning (*partiality*, *approximation*, *proximity*) and shown to be technically more general than PLC [6].

Context reasoning mechanisms have already been used in practice in various distributed reasoning applications, and are expected to play a significant role in the development of the next generation AI applications. Prominent examples include (a) the CYC common sense knowledge base designed as a collection of interrelated partial "*microtheories*" [7,8]; (b) languages that have been developed for the expression of *contextualized ontologies*, such as Distributed Description Logics [9] and C-OWL [10]; and (c) the MCS-based agent architectures of [11] and [12].

Recent studies proposed adding non-monotonic features to MCS to address incomplete, uncertain or ambiguous context information. Two representative examples are: (a) the non-monotonic rule-based MCS framework [13], which supports default negation in the mapping rules; and (b) the multi-context variant of Default Logic, *ConDL* [14], which models bridge relations between different contexts as *default rules,* in order to handle the case of mutually inconsistent context information. A third approach [15], proposed a non-monotonic extension to MCS, which supports cases of missing or inaccurate local context knowledge by representing contexts as local theories of Defeasible Logic. To handle potential inconsistencies caused by the interaction of contexts, the MCS model was extended with defeasible mapping rules and with a total preference ordering on the system contexts. A semantic characterization of the model using *arguments* and an operational model in the form of a distributed algorithm for query evaluation were provided. This approach was applied to the Ambient Intelligence domain, which is characterized by open and dynamic environments, distributed context information and imperfection of the available knowledge.

In this paper, we extend the work of [15] to cover cases of incomplete preference information. So far, perfect preference information was assumed in the sense that (a) all agents need to be compared to each other, and (b) that each agent has a total preference relation on all fellow agents. However, this requirement may not by realistic in uncertain environments. In Ambient Intelligence environments an agent may collect information of various types from several different information sources, which cannot be compared in terms of confidence. It is not, for example, rational to compare a localization service with a weather forecast service. To handle such cases, we assume incomplete preference information, modeled as partial ordering on the system contexts, and modify accordingly the argumentation semantics and the distributed query evaluation algorithm. Using an illustrative scenario from Social Networking, we demonstrate the main features of this new approach.

The rest of the paper is structured as follows. Section 2 describes a scenario that highlights the requirements for reasoning with imperfect context and preference information. Section 3 describes the context representation model, Section 4 provides an argumentation system for the intended contextual reasoning in the model, while Section 5 provides distributed query evaluation algorithm that is sound and complete w.r.t. the argumentation system. Section 6 discusses related work, while section 7 concludes and proposes directions for future work.

2 Illustrative Scenario

This scenario involves a student, Adam, using a cell phone and takes place at University of Crete, Greece. An SMS is received by Adam's cell phone at 10 am from a multiplex cinema to inform him that a new action movie is available at 10pm that night at the open air cinema by the beach. Based on Adam's profile information that it carries and on knowledge about Adam's context, the mobile phone must determine whether Adam should be informed about the movie.

Adam' profile contains rules dictating activity notification based on information involving his schedule, weather, personal interests and university obligations.

Adam's preferences about movies are included in his profile. Adam has also specified two web sites – meteo.gr and poseidon.gr – as external sources for information about weather conditions. Meteo.gr describes general weather conditions, while poseidon.gr specializes more on wind reports. For class related information, the mobile phone contacts the University of Crete web server that retains the schedule of all classes. Finally, the external sources used for information about movies are IMDB and Rotten Tomatoes. Adam has also specified that he considers poseidon.gr more trustworthy than meteo.gr, while when it comes to movie-related information, he prefers IMDB to Rotten Tomatoes.

For the specific scenario, we assume that Adam is interested in action movies. Knowledge imported from meteo.gr indicates good weather conditions for an outdoor activity, while poseidon.gr claims the opposite regarding winds. Based on class information imported from the University of Crete web server, there is no conflict for the date and time in question. Finally, we assume that IMDB suggests it's a good movie, while Rotten Tomatoes holds the opposite view.

3 Context Representation Model

We model a Multi-Context System C as a collection of distributed context theories C_i: A context is defined as a tuple of the form(V_i, R_i, T_i) where V_i is the vocabulary used by C_i, R_i is a set of rules, and T_i is a preference relation on C.

V_i is a set of positive and negative literals. If q_i is a literal in V_i, $\sim q_i$ denotes the complementary literal, which is also in V_i. If q_i is a positive literal p then $\sim q_i$ is $\neg p$; and if q_i is $\neg p$, then $\sim q_i$ is p. We assume that each context uses a distinct vocabulary.

R_i consists of two sets of rules: the set of local rules and the set of mapping rules. The body of local rules is a conjunction of local literals (literals that are contained V_i), while their head contains a single local literal. There are two types of local rules:

– Strict rules, of the form: $r_i^l: a_i^1, a_i^2, \dots a_i^{n-1} \rightarrow a_i^n$. They express local knowledge and are interpreted in the classical sense: whenever the literals in the body of the rule are strict consequences of the local theory, then so is the literal in the head of the rule. Local rules with empty body denote factual knowledge.
– Defeasible rules, of the form: $r_i^d: b_i^1, b_i^2, \dots b_i^{n-1} \Rightarrow b_i^n$. They are used to express uncertainty, in the sense that a defeasible rule cannot be applied to support its conclusion if there is adequate contrary evidence.

Mapping rules associate local literals with literals from the vocabularies of other contexts (foreign literals). The body of each such rule is a conjunction of local and foreign literals, while its head contains a single local literal: $r_i^m: a_i^1, a_j^2, \dots a_k^{n-1} \Rightarrow a_i^n$. r_i^m associates local literals of C_i (e.g. a_i^1) with local literals of C_j (a_j^2), C_i(a_k^{n-1}) and possibly other contexts. a_i^n is a local literal of the theory that has defined r_i^m (C_i).

Finally, each context C_i defines a preference order. Formally it is defined as a partial order T_i on C, to express its confidence in the knowledge it imports from other contexts. Represented as a set: $T_i = \{ L_j \}$, where L_j is a list representing a path of the partial order tree in which when a context C_k precedes another context C_l then C_k is preferred over C_l.

Example. The scenario described in Section 2 can be represented by the following context theories. Rules r_{11} to r_{110} constitute the context theory of the mobile phone (represented as context C_1), rules r^l_{21} and r^l_{22} constitute the context theory of meteo.gr (C_2), while rules r^l_{31}, r^l_{41}, r^l_{51} and r^l_{61} constitute respectively the context theories of poseidon.gr (C_3), the university web server (C_4), imdb.com (C_5) and Rotten Tomatoes (C_6). Note that through rules r_{16} to r_{110}, the mobile phone imports context information from the available external sources. T_1 represents the preference ordering defined by C_1.

– r^d_{11}: $\mathrm{goodWeather}_1, \mathrm{freeSchedule}_1, \mathrm{likesMovie}_1, \mathrm{goodMovie}_1 \Rightarrow \mathrm{activity}_1$
– r^d_{12}: $\mathrm{badWeather}_1 \Rightarrow \neg\mathrm{activity}_1$
– r^d_{13}: $\mathrm{badMovie}_1 \Rightarrow \neg\mathrm{activity}_1$
– r^l_{14}: $\mathrm{actionMovie}_1 \rightarrow \mathrm{likesMovie}_1$
– r^l_{15}: $\mathrm{actionMovie}_1$
– r^m_{16}: $sunny_2, high\_temp_2 \Rightarrow \mathrm{goodWeather}_1$
– $\mathrm{r}^{\mathrm{m}}_{17}$: $\mathrm{windy}_3 \Rightarrow \mathrm{badWeather}_1$
– $\mathrm{r}^{\mathrm{m}}_{18}$: $\neg\mathrm{scheduledlesson}_4 \Rightarrow \mathrm{freeSchedule}_1$
– $\mathrm{r}^{\mathrm{m}}_{19}$: $\mathrm{highratedMovie}_5 \Rightarrow \mathrm{goodMovie}_1$
– $\mathrm{r}^{\mathrm{m}}_{110}$: $\mathrm{lowratedMovie}_6 \Rightarrow \mathrm{goodMovie}_1$
– $T_1 = \{[C_3, C_2], [C_5, C_6]\}$

– r^l_{21}: $\rightarrow sunny_2$ r^l_{41}: $\rightarrow \neg scheduledlesson_4$
– r^l_{22}: $\rightarrow high\_temp_2$ r^l_{51}: $\rightarrow highratedMovie_5$
– r^l_{31}: $\rightarrow windy_3$ r^l_{61}: $\rightarrow lowratedMovie_6$

4 Argumentation Semantics

The argumentation framework described in this section extends the argumentation semantics of Defeasible Logic [19] with the notions of distribution of the available information, and preference among system contexts. It modifies the argumentation framework in [15] by relaxing the requirements of total preference order.

The framework uses arguments of local range, in the sense that each one contains rules of a single context only. Arguments of different contexts are interrelated in the Support Relation (Definition 1) through mapping rules. The relation contains triples; each triple contains a proof tree for a literal of a context using the rules of the context. The proof tree may contain either only local rules, or both local and mapping rules. In the second case, for a triple to be contained in the Support Relation, similar triples for the foreign literals of the triple must have already been obtained. We should also note that for sake of simplicity we assume that there are no loops in the local context theories, and thus proof trees are finite. However the global knowledge base may contain loops caused by mapping rules, which associate different context theories.

Definition 1. Let $\mathrm{C} = \{\mathrm{C_i}\}$ be a Defeasible MCS. The Support Relation of C ($\mathrm{SR_C}$) is the set of all triples of the form $(\mathrm{C_i}, \mathrm{PT_{p_i}}, \mathrm{p_i})$, where $\mathrm{C_i} \in \mathrm{C}, \mathrm{p_i} \in \mathrm{V_i}$, and $\mathrm{PT_{p_i}}$, is the proof tree for $\mathrm{p_i}$ based on the set of local and mapping rules of $\mathrm{C_i}$. $\mathrm{PT_{p_i}}$, is a tree with nodes labeled by literals such that the root is labeled by p_i, and for every node with label q:

1. $If\ q \in V_i\ and\ a_1, \dots, a_n\ label\ the\ children\ of\ q\ then$
$-\ If\ \forall a_i (i = 1, \dots, n): a_i \in V_i\ then\ there\ is\ a\ local\ rule\ r_i \in C_i\ with\ body\ a_1, \dots, a_n and\ head$
$-\ If\ \exists a_j\ such\ that\ a_j \notin V_i\ then\ there\ is\ a\ mapping\ rule\ r_i \in C_i\ with\ body\ a_1, \dots, a_n and\ head\ q$
2. $If\ q\ \in V_j \neq V_i, then\ this\ is\ a\ leaf\ node\ of\ the\ tree\ and\ there\ is\ a\ triple\ of\ the\ form\ (C_j, PT_q, q\ )\ in\ SR_C$
3. $The\ arcs\ in\ a\ proof\ tree\ are\ labeled\ by\ the\ rules\ used\ to\ botain\ them.$

Definition 2. $An\ argument\ A\ for\ a\ literal\ p_i is\ a\ triple\ (C_i, PT_{p_i}, p_i)\ in\ SR_C.$

Any literal labelling a node of PT_{p_i} is called a *conclusion* of A. However, when we refer to the *conclusion* of A, we refer to the literal labelling the root of PT_{p_i} (p_i). We write r ∈ A to denote that rule r is used in the proof tree of A.

The definition of *subarguments* is based on the notion of subtrees.

Definition 3. *A (proper) subargument of A is every argument with a proof tree that is (proper) subtree of the proof tree of A.*

Based on the literals contained in their proof tree, arguments are classified to local arguments and *mapping* arguments. Based on the type of the rules that they use, local arguments are either *strict* local arguments or *defeasible* local arguments.

Definition 4. *A local argument of context C_i is an argument that contains only local literals of C_i. If a local argument A contains only strict rules, then A is a strict local argument; otherwise it is a defeasible local argument. A is mapping argument if its proof tree contains at least one foreign literal.*

Definition 5. *$Args_{C_i}$ is the set of arguments derived from context C_i. $Args_C$ is the set of all arguments in C: $Args_C = \bigcup Args_{C_i}$.*

The conclusions of all strict local arguments in $Args_{C_i}$ are logical consequences of C_i. Distributed logical consequences are derived from a combination of local and mapping arguments in $Args_C$. In this case, we should also consider conflicts between competing rules, which are modelled as attacks between arguments, and preference orderings, which are used in our framework to compare mapping arguments.

Definition 6. *An argument A attacks a defeasible local or mapping argument B at p_i, if p_i is a conclusion of B, $\sim p_i$ is a conclusion of A, and the subargument of B with conclusion p_i is not a strict local argument.*

Definition 7. *An argument A defeats a defeasible or mapping argument B at p_i, if A attacks B at p_i, and for the subarguments of A, A' with conclusion $\sim p_i$, and of B, B' with conclusion p_i, it holds that either:*

1. A' is a local argument of C_i or
2. $B' is\ a\ mapping\ argument\ of\ C_i$ and
$\exists b_l \in B : \forall a_k \in A, \exists L_j \in T_i : C_l, C_k \in L_j\ and\ C_l\ precedes\ C_k in\ L_j$

We assume that a literal a_k is defined in context and $C_k \neq C_i$. Partial orders are acyclic, therefore it cannot happen that A defeats B and B defeats A based on condition 2.

To link arguments through the mapping rules that they contain, we introduce in our framework the notion of *argumentation line*.

Definition 8. *A*n argumentation line A_L for a literal p_i is a sequence of arguments in $Args_C$, constructed in steps *as follows:*

- *In the first step add in* A_L *one argument for* p_i.
- *In each next step, for each distinct literal* q_j *labeling a leaf node of the proof trees of the arguments added in the previous step, add one argument with conclusion* q_j, *with the following restriction.*
- *An argument B for a literal* q_j *can be added in* A_L *only if there is no argument* $D \neq B$ *for* q_j *already in* A_L.

The argument for p_i added in the first step is called the head argument of A_L. If the number of steps required to build an A_L is finite, then A_L is a finite argumentation line. Infinite argumentation lines imply loops in the global knowledge base. Arguments contained in infinite argumentation lines participate in *attacks* against counter-arguments but may not be used to support the conclusion of their argumentation lines.

The notion of supported argument is meant to indicate when an argument may have an active role in proving or preventing the derivation of a conclusion.

Definition 9. *An argument A is supported by a set of arguments S if:*

- *every proper subargument of A is in S and*
- *there is a finite argumentation line* A_L *with head A, such that every argument in* $A_L - \{A\}$ *is in S*

That an argument *A* is *undercut* by a set of arguments *S* means that we can show that some premises of *A* cannot be proved if we accept the arguments in *S*.

Definition 10. *A defeasible local or mapping argument A is undercut by a set of arguments S if for every argumentation line* A_L *with head A: there is an argument B, such that B is supported by S, and B defeats a proper subargument of A or an argument in* $A_L - \{A\}$.

The definition of *acceptable* arguments that follows is based on the definitions given above. Intuitively, that an argument *A* is *acceptable* w.r.t. *S* means that if we accept the arguments in S as valid arguments, then we feel compelled to accept *A* as valid.

Definition 11. *An argument A is acceptable w.r.t. a set of arguments S if:*

1. *A is a strict local argument; or*
2. *(a) A is supported by S and*
   (b) every argument defeating A is undercut by S

Based on the concept of acceptable arguments, we proceed to define justified arguments and justified literals.

Definition 12. *Let C be a MCS.* J_i^C *is defined as follows:*

- $J_0^C = \emptyset$;
- $J_{i+1}^C = \{A\ Args_C \mid A\ is\ acceptable\ w.r.t. J_i^C\}$

The set of *justified arguments* in a MCS C is $JArgs^C = \bigcup_{i=1}^{\infty} J_i^C$. A literal p_i is *justified* if it is the conclusion of an argument in $JArgs^C$. That an argument A is justified means that is resists every reasonable refutation. That a literal p_i is justified, it actually means that it is a logical consequence of *C*.

Finally, we also introduce the notion of *rejected arguments* and *rejected literals* for the characterization of conclusions that do not derive from *C*. That an argument is rejected by sets of arguments *S* and *T* means that either it is supported by arguments in *S*, which can be thought of as the set of already rejected arguments, or it cannot overcome an attack from an argument supported by *T*, which can be thought of as the set of justified arguments.

Definition 13. *An argument A is rejected by sets of arguments S, T when:*

1. *A is not a strict local argument, and either*
2. *(a) a proper subargument of A is in S; or*
   (b) A is defeated by an argument supported by T; or
   (c) for every argumentation line A_L*with head A there exists an argument* $A' \in A_L - \{A\}$*, such that either a subargument of* A' *is in S; or* A' *is defeated by an argument supported by T.*

Based on the definition of rejected arguments, R_i^C is defined as follows:

Definition 14. *Let C be a MCS, and* $JArgs^C$*the set of justified arguments in C.* J_i^C *is defined as follows:*

- $R_0^C = \emptyset$;
- $R_{i+1}^C = \{A \in Args_C \mid A\ is\ rejected\ by\ R_i^C, JArgs^C\}$

The set of *rejected arguments* in a MCS C is $RArgs^C = \bigcup_{i=1}^{\infty} R_i^C$. A literal p_i is *rejected* if there is no argument in $Args^C - RArgs^C$ that supports p_i. That a literal is rejected means that we are able to prove that it is not a logical consequence of *C*.

Lemmata 1-3 describe some formal properties of the framework. Their proofs are omitted due to space limitations.

Lemma 1. *The sequences* J_i^C *and* $R_i^C(T)$ *are monotonically increasing.*

Lemma 2. *In a Multi-Context System C:*

- *No argument is both justified and rejected.*
- *No literal is both justified and rejected.*

Lemma 3. *If the set of justified arguments of C,* $JArgs_C$ *contains two arguments with complementary conclusions, then both arguments are local arguments of the same context.*

Example (continued). The arguments that are derived from $MCS\ C$ of the previous section are depicted in Figure 1 along with their subarguments.

J_0^C contains no arguments, while J_1^C contains the strict local arguments of the system; namely $A_1', A_1'', A_{21}, A_{22}, A_4, A_5, B_3, D_6$, where A_1' and A_1'' are the strict local subarguments of A_1 with conclusion $\mathrm{likesMovie}_1$ and $\mathrm{actionMovie}_1$ respectively.

J_2^C additionally contains A_1's subarguments with conclusions $\mathrm{goodWeather}_1$, freeSchedule_1, $\mathrm{goodMovie}_1$ as well as B_1's subargument with conclusion $\mathrm{badWeather}_1$ and D_1's subargument with conclusion $\mathrm{badMovie}_1$. J_2^C actually constitutes the set of justified arguments in C ($\mathrm{JArgs}^C = J_2^C$), as there is no argument that can be added in the next steps of J_i^C.

On the other hand, R_0^C (JArgs^C) contains no arguments, while R_1^C (JArgs^C), which equals $\mathrm{RArgs}^C(\mathrm{JArgs}^C)$ contains A_1, B_1 and D_1, as each of them is defeated by a justified argument. Hence $\mathrm{activity}_1$ and $\neg\mathrm{activity}_1$ are rejected literals since every argument supporting them is rejected.

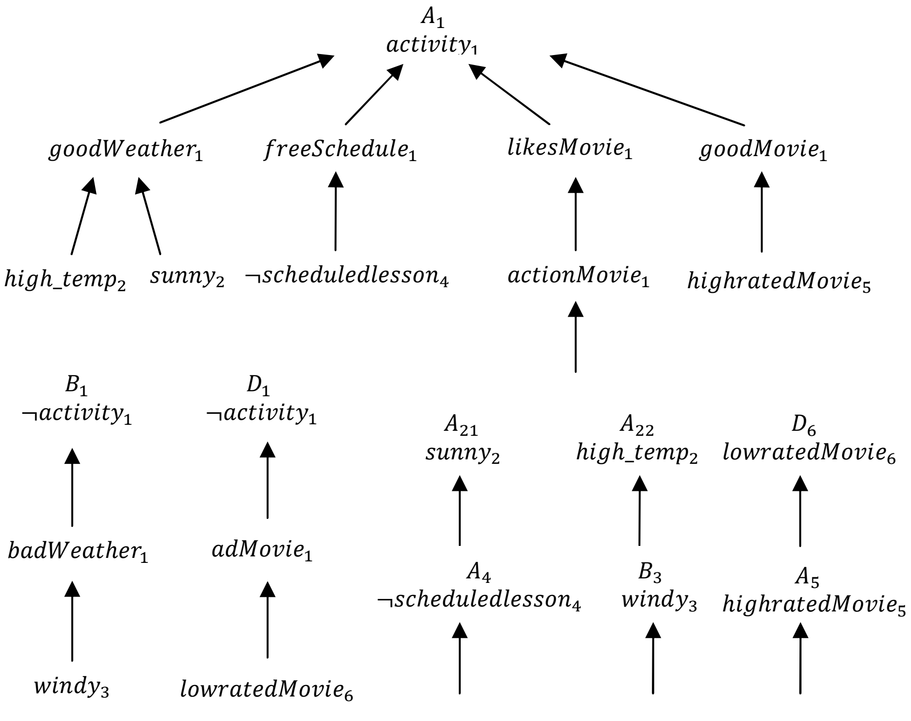


Fig. 1. Arguments in the scenario

5 Distributed Algorithm for Query Evaluation

P2P_DR is a distributed algorithm for query evaluation that implements the proposed argumentation framework. The specific problem that it deals with is: *Given a MCS C, and a query about literal p_i issued to context C_i, compute the truth value of p_i.* For an arbitrary literal p_i, *P2P_DR* returns one of the following values: (a) *true*; indicating that p_i is a logical consequence of *C*; (b) *false*; indicating that p_i is not a logical consequence of C; or (c) *undefined*; indicating that based on C, we cannot derive neither *true* nor *false* as a truth value for p_i. This algorithm differs from the one that implemented our original approach [15] only in the *Stronger* function which implements the comparison between two sets of literals. Below, we describe the algorithm verbally, and give the code of the *Stronger* function that implements the modification to our original approach.

5.1 Algorithm Description

P2P_DR proceeds in four main steps. In the first step, *P2P_DR* determines whether p_i, or its negation $\sim p_i$ are consequences of the strict local rules of C_i, returning *true/false* respectively as an answer for p_i and terminates.

In the second step, *P2P_DR* calls *Support* to determine whether there are applicable and unblocked rules with head p_i. We call *applicable* those rules that for all literals in their body *P2P_DR* has computed *true* as their truth value, while *unblocked* are the rules that for all literals in their body *P2P_DR* has computed either *true* or *undefined* as their truth value. *Support* returns two data structures for p_i: (a) the Supportive Set of pi (SS_{p_i}), which is the set of foreign literals used in the most preferred (according to T_i) chain of applicable rules for p_i ; and (b) the Blocking Set of p_i (BS_{p_i}), which is the set of foreign literals used in the most preferred chain of unblocked rules for p_i (BS_{p_i}). If there is no unblocked rule for p_i ($BS_{p_i} = \emptyset$), the algorithm returns *false* as an answer and terminates. Similarly, in the third step, *P2P_DR* calls *Support* to compute the respective constructs for $\sim p_i$ ($SS_{\sim p_i}, BS_{\sim p_i}$).

In the last step, *P2P_DR* uses the constructs computed in the previous steps and the preference order T_i, to determine the truth value of p_i. In case there is no unblocked rule for $\sim p_i$ ($BS_{\sim p_i} = \emptyset$), or SS_{p_i} is computed by *Stronger* to be *stronger* than $BS_{\sim p_i}$, *P2P_DR* returns *true* as an answer for p_i. That SS_{p_i} is *stronger* than BS_{p_i} means that the chains of applicable rules for p_i involve information from contexts that are preferred by C_i to the contexts that are involved in the chain of unblocked rules for $\sim p_i$. In case there is at least one applicable rule for $\sim p_i$, and BS_{p_i} is *not stronger* than $SS_{\sim p_i}$, *P2P_DR* returns *false* as an answer for p_i. In any other case, the algorithm returns *undefined.*

The *Stronger*(A, B, T_i) function computes the *strongest* between two sets of literals, A and B according to the preference order T_i.

```
Stronger $(A, B, T_i)$
1: if $\exists b_l \in B : \forall a_k \in A, \exists L_j \in T_i: C_l, C_k \in L_j$ and $C_l$ precedes $C_k$ in $L_j$ then
2:         Stronger = A
3: else if $\exists a_k \in A : \forall b_l \in B, \exists L_j \in T_i: C_k, C_l \in L_j$ and $C_k$ precedes $C_l$ in $L_j$ then
4:         Stronger = B
5: else
6:         Stronger = None
7: return Stronger
```

Example (continued). Given a query about $activity_1$, *P2P_DR* proceeds as follows. At first, it fails to compute an answer based only on C_1's local theory from which it derives only likesMovie_1 using strict local rules r^1_{14} and r^1_{15}. Next step is to use mapping rules

r^m_{16}, r^m_{18} and r^m_{19} to compute an answer for $\mathrm{goodWeather}_1, \mathrm{freeSchedule}_1$ and $\mathrm{goodMovie}_1$, based on strict local rules of C_2, C_4 and C_5 respectively. These rules are applicable and support $\mathrm{activity}_1$, as are rules r^d_{12} and r^d_{13} supporting the opposite $\neg\mathrm{activity}_1$. Their respective computed sets are the following: $SS_{\mathrm{activity}} = \{\ \mathrm{high}_{\mathrm{temp}_2}, \mathrm{sunny}_2, \neg\mathrm{scheduledlesson}_4, \mathrm{highratedMovie}_5\}$ and the $BS_{\sim\mathrm{activity}} = \{\mathrm{windy}_3, \mathrm{lowratedMovie}_6\}$.

Based on T_1, *Stronger* returns *none* and eventually *P2P_DR* returns false for $activity_1$. The same result would occur for a query about $\neg\mathrm{activity}_1$.

5.2 Properties of the Algorithm

Below, we describe some formal properties of *P2P_DR* regarding its termination, soundness and completeness w.r.t. the argumentation framework, communication and computational load. Here, we give an overview of the results. The proofs for the following propositions are omitted due to space limitations. Prop. 1 is a consequence of the cycle detection process within the algorithm.

Proposition 1. (Termination) *The algorithm is guaranteed to terminate returning either a positive or a negative answer for the queried literal.*

Prop. 2 associates the answers produced by *P2P_DR* with the concepts of justified and rejected arguments.

Proposition 2. For *a Multi-Context System C* and a literal p_i in *C, P2_ DR* returns:

1. $\mathrm{Ans}_{p_i} = \ true\ iff\ p_i\ is\ justified$
2. $\mathrm{Ans}_{p_i} = false\ iff\ p_i is\ rejected\ by\ JArgs^C$
3. $\mathrm{Ans}_{p_i} = \ undfined\ iff\ p_i is\ neither\ justified\ nor\ rejected\ by\ JArgs^C$

Prop. 3 is consequence of two states that we retain for each context, keeping track of the incoming and outgoing queries of the context. The worst case is that all contexts have defined mappings that contain literals from all other contexts, and the evaluation of the query involves all mappings defined in the system.

Proposition 3. (Number of Messages) *The total number of messages exchanged between the system contexts for the evaluation of a query is* $O(n^3\ x\ n_l^2)$, *where n stands for the total number of system contexts, and* n_l *stands for the number of literals a context may define.*

6 Related Work

The argumentation framework that we propose in this paper belongs to the broader family of *preference-based argumentation systems*. The main feature of such systems is that the comparison between arguments (e.g. between an argument and its counter-arguments) is enabled by a preference relation, which is either implicitly derived from elements of the underlying theory, or is explicitly defined on the set of arguments. Such systems can be classified into four categories. In the first category, which includes the works of [16] and [17], the preference relation takes into account the internal structure of arguments, and arguments are compared in terms of specificity. The second category includes systems in which preferences among arguments are derived from a priority relation on the rules in the underlying theory (e.g. [18,19]). In Value Based Argumentation Frameworks, the preference ordering on the set of arguments is derived from a preference ordering over the values that they promote (e.g. [20,21]). Finally, the abstract argumentation frameworks proposed by Amgoud and her colleagues ([22,23]) assume that preferences among arguments are induced by a preference relation defined on the underlying belief base.

Our argumentation framework is an extension of the framework of Governatori et al. [19], which is based on the grounded semantics of Dung's abstract argumentation framework [24] to provide an argumentative characterization of Defeasible Logic. In our framework, preferences are derived both from the structure of arguments – arguments that use strict local rules are considered stronger than those that use defeasible local or mapping rules - and from a preference ordering on the information sources (contexts). Our approach also shares common ideas with [22], which first introduced the notion of contextual preferences (in the form of several preorderings on the belief base), to take into account preferences that depend upon a particular context. The main differences are that in our case, these orderings are applied to the contexts themselves rather than directly to a set of arguments, and that we use a distributed underlying knowledge base.

7 Conclusions

The paper studies the problem of reasoning with imperfect context and preference information in Multi-Context Systems. In the proposed framework, uncertainty in the local theories is modelled using defeasible local theories, while conflicts that may arise from the interaction of contexts through mappings are resolved using contextual preference information. To support missing preference information, we use partial preference ordering on the set of contexts. The paper also provides an argumentation-based semantic characterization of the model, and a distributed algorithm for query evaluation that implements the argumentation framework.

Our ongoing work involves: (a) studying the complexity of the proposed algorithm, based on the complexity results of our original approach [15]; (b) studying alternative methods for conflict resolution, which differ in the way that agents evaluate the imported context information; and (c) implementing real-world applications of our approach in Ambient Intelligence environments, such as those described in [24,25], but also in other domains with similar requirements such as Social Networks and the Semantic Web.

References

1. McCarthy, J.: Generality in Artificial Intelligence. Communications of the ACM 30(12), 1030–1035 (1987)
2. Buvac, S., Mason, I.A.: Propositional Logic of Context. In: AAAI, pp. 412–419 (1993)
3. McCarthy, J., Buvac, S.: Formalizing Context (Expanded Notes). In: Aliseda, A., van Glabbeek, R., Westerstahl, D. (eds.) Computing Natural Language, pp. 13–50. CSLI Publications, Stanford (1998)
4. Giunchiglia, F., Serafini, L.: Multilanguage hierarchical logics, or: how we can do without modal logics. Artificial Intelligence 65(1) (1994)

5. Ghidini, C., Giunchiglia, F.: Local Models Semantics, or contextual reasoning=locality+ compatibility. Artificial Intelligence 127(2), 221–259 (2001)
6. Serafini, Bouquet, P.: Comparing formal theories of context in AI. Artificial Intelligence 155(1-2), 41–67 (2004)
7. Guha, R.: Contexts: a formalization and some applications. PhD thesis, Stanford, CA, USA (1992)
8. Lenat, D., Guha, R.: Building Large Knowledge-Based Systems; Representation and Inference in the Cyc Project. Addison-Wesley Longman Publishing Co. Inc., Boston (1989)
9. Borgida, A., Serafini, L.: Distributed Description Logics: Assimilating Information from Peer Sources. Journal of Data Semantics 1, 153–184 (2003)
10. Bouquet, P., Giunchiglia, F., van Harmelen, F., Serafini, L., Stuckenschmidt, H.: C-OWL: Contextualizing Ontologies. In: Fensel, D., Sycara, K., Mylopoulos, J. (eds.) ISWC 2003. LNCS, vol. 2870, pp. 164–179. Springer, Heidelberg (2003)
11. Parsons, S., Sierra, C., Jennings, N.R.: Agents that reason and negotiate by arguing. Journal of Logic and Computation 8(3), 261–292 (1998)
12. Sabater, J., Sierra, C., Parsons, S., Jennings, N.R.: Engineering Executable Agents using Multi-context Systems. Journal of Logic and Computation 12(3), 413–442 (2002)
13. Roelofsen, F., Serafini, L.: Minimal and Absent Information in Contexts. In: IJCAI, pp. 558–563 (2005)
14. Brewka, G., Roelofsen, F., Serafini, L.: Contextual Default Reasoning. In: IJCAI, pp. 268–273 (2007)
15. Bikakis, A., Antoniou, G.: Contextual Argumentation in Ambient Intelligence. In: Erdem, E., Lin, F., Schaub, T. (eds.) LPNMR 2009. LNCS, vol. 5753, pp. 30–43. Springer, Heidelberg (2009); An extended version was accepted by IEEE Transactions on Knowledge and Data Engineering
16. Simari, G.R., Loui, R.P.: A Mathematical Treatment of Defeasible Reasoning and its Implementation. Artificial Intelligence 53(2-3), 125–157 (1992)
17. Stolzenburg, F., García, A.J., Chesñevar, C.I., Simari, G.R.: Computing Generalized Specificity. Journal of Applied Non-Classical Logics 13(1), 87–113 (2003)
18. Prakken, H., Sartor, G.: Argument-Based Extended Logic Programming with Defeasible Priorities. Journal of Applied Non-Classical Logics 7(1) (1997)
19. Governatori, G., Maher, M.J., Billington, D., Antoniou, G.: Argumentation Semantics for Defeasible Logics. Journal of Logic and Computation 14(5), 675–702 (2004)
20. Bench-Capon, T.: Persuasion in Practical Argument Using Value-based Argumentation Frameworks. Journal of Logic and Computation 13, 429–448 (2003)
21. Kaci, S., van der Torre, L.: Preference-based argumentation: Arguments supporting multiple values. Internation Journal of Approximate Reasoning 48(3), 730–751 (2008)
22. Amgoud, L., Parsons, S., Perrussel, L.: An Argumentation Framework based on contextual Preferences. In: International Conference on Formal and Applied and Practical Reasoning (FAPR 2000), pp. 59–67 (2000)
23. Amgoud, L., Cayrol, C.: A Reasoning Model Based on the Production of Acceptable Arguments. Annals of Mathematic and Artificial Intelligence 34(1-3), 197–215 (2002)
24. Dung, P.M.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artif. Intell. 77, 321–357 (1995)
25. Bikakis, A., Antoniou, G.: Distributed Defeasible Contextual Reasoning in Ambient Computing. In: Aarts, E., Crowley, J.L., de Ruyter, B., Gerhäuser, H., Pflaum, A., Schmidt, J., Wichert, R. (eds.) AmI 2008. LNCS, vol. 5355, pp. 308–325. Springer, Heidelberg (2008)
26. Antoniou, G., Papatheodorou, C., Bikakis, A.: Distributed Reasoning about Context in Ambient Intelligence Environments: A Report from the Filed. In: Proceedings of the 12th International Conference on the Principles of Knowledge Representation and Reasoning, KR 2010 (accepted 2010)

Paid Content a Way to Electronic Knowledge-Based Economy

Wojciech Cellary

Department of Information Technology, Poznan University of Economics
Mansfelda 4, 60-854 Poznan, Poland
cellary@kti.ue.poznan.pl

Abstract of a Keynote Speech

Nowadays, two the most significant concepts determining the future of the world are: information society and knowledge-based economy. Information society may be defined as a community whose collective life is organized by the wide use of the information and communication technologies, and which economy is based on knowledge. An economy is based on knowledge if the market value of dominating products and services depends mostly on knowledge, instead of resources, energy, or physical work.

This keynote speech deals with the dilemma followed from the double character of knowledge. On the one hand, knowledge is an individual and social good determining quality of life. On the other hand, knowledge has to be treated as an economical good, a funding concept of the knowledge-based economy. The dilemma is: should knowledge be for free or paid? It is argued in the speech that achievement of information society without knowledge-based economy is impossible. In countries focused on education, about 50% of young people study at universities. This means that after unavoidable, natural replacement of the old generation by the young one, a half of society would like to earn a living providing knowledge-based services, then such activity must be a part of economy. As totally unrealistic should be evaluated an economic model in which non-educated part of the society works physically in the private sector and pays taxes to finance public sector, while the educated part of the society is paid from taxes to provide "free of charges" knowledge-based services. In addition, a characteristic feature of knowledge is a positive feedback: more knowledge acquired – more demand for new knowledge. Due to cognitive curiosity of highly-skilled people, their demand for new knowledge is bigger than this of non-educated people, even though objectively, a non-educated person should be interested in receiving more knowledge to improve his/her life.

The more knowledge the society will be given for free – which means in reality financed from public funds – the more knowledge well-educated part of the society will demand. As a consequence, it will be necessary to increase the fiscal charges of non-educated part of the society. Such economic system – as every one based on the positive feedback, a phenomenon well known from the control theory – is unstable and must fail. That is why the knowledge-based economy

needs a stable economic model in which knowledge will be first of all considered as an economic good – a commodity – an object of purchase and sale.

Economical activity with respect to knowledge – like in every healthy economic model – should generate profit for enterprises, and should be taxed to bring income to the state budget enabling realization of social goals. To conclude, the more knowledge workers will earn a living providing paid knowledge-based services as business activities, the more profits for the whole society.

The necessity of knowledge-based economy development does not exclude public aid and realization of its part in the public sector. Knowledge financed from the public sources is motivated by the need of sustainable development of economy as a condition of social welfare. However, a question arises: what can be realized in the public sector to prevent knowledge-based economy from damages? The proposed answer is: only precisely defined part of the knowledge sector: first, knowledge that does not develop any more, and knowledge necessary to bring new social groups of people to the knowledge-based economy. For economic reason, it is purposeful to finance every action aiming at extension of the knowledge sector market from public funds. The argument for such financing is that people without certain basic knowledge cannot play an active role on the knowledge market. From the economic point of view, it is also acceptable to finance the "knowledge without development", i.e., a canon of knowledge. Such financing does not stop knowledge-based economy development, because this kind of knowledge does not develop anymore.

A completely different threat for knowledge-based economy development is knowledge privatization. Privatization of knowledge can also stop development of the knowledge-based economy. First, for economic reasons: potential users of knowledge cannot afford to buy it. Second, for lack of permission of the knowledge owners to use the knowledge they own. The second reason seems to be irrational from the economic point of view. However, in many real circumstances, an enterprise decides to launch a new knowledge-based product or service not in the earliest possible moment, but in the most convenient for it, i.e., when economic conditions are considered the best. The owner of some piece of knowledge may do not want to use it to develop new products or services, because he/she does not want the new products or services to compete with its current ones, whose sale is still going well. Moreover, such owner of a piece of knowledge may prevent his/her competitors from exploiting that piece of knowledge, even if they have acquired the same piece of knowledge independently.

As follows from considerations presented in this speech, it is very important to find an appropriate economic model based on a right trade-off between social and economic values. Free access to knowledge for everyone financed from public funds is not a right model, because it does not contribute to development.

Reference

1. Cellary, W.: Paid content a way to electronic knowledge-based economy. In: Hansen, H.R., Karagiannis, D., Fill, H.-G. (eds.) 1st CEE Symposium on Business Informatics, pp. 11–18. Austrian Computer Society, Vienna (2009)

DBTech EXT: Education and Hands-On Training for the Database Professional*

Martti Laiho[1], Dimitris A. Dervos[2], José F. Aldana-Montes[3], and Fritz Laux[4]

[1] Haaga-Helia University of Applied Sciences, Helsinki, Finland
martti.laiho@haaga-helia.fi
[2] Alexander Technology Educational Institute, Thessaloniki, Greece
dad@it.teithe.gr
[3] Universidad de Málaga, Málaga, España
jfam@lcc.uma.es
[4] Reutlingen University, Reutlingen, Germany
Friedrich.Laux@Reutlingen-University.DE

Abstract. In this presentation the audience will be: (a) introduced to the aims and objectives of the DBTechNet initiative, (b) briefed on the DBTech EXT virtual laboratory workshops (VLW), i.e. the educational and training (E&T) content which is freely available over the internet and includes vendor-neutral hands-on laboratory training sessions on key database technology topics, and (c) informed on some of the practical problems encountered and the way they have been addressed. Last but not least, the audience will be invited to consider incorporating some or all of the DBTech EXT VLW content into their higher education (HE), vocational education and training (VET), and/or lifelong learning/training type course curricula. This will come at no cost and no commitment on behalf of the teacher/trainer; the latter is only expected to provide his/her feedback on the pedagogical value and the quality of the E&T content received/used.

Keywords: DBTechNet, vendor-neutral educational and training content, virtual laboratory workshops, lifelong learning, e-learning.

1 Introduction

Established in 1997, the DBTechNet initiative [1] has set forth a plan to address the lack of knowledgeable and well-trained database professionals in the European labour market: a low productivity problem that has led to failed and delayed application development projects in the industry. The precondition first to be met was the formation of a team of HE teachers who all were: (a) enthusiastic in sharing the common EU vision, (b) willing to contribute their specialised

* This presentation is the result of collaborative work undertaken along the lines of the DBTechNet Consortium. All the co-authors participate in DBTech EXT, a project partially funded by the EU LLP Transversal Programme (Project Number: 143371-LLP-1-2008-1-FI-KA3-KA3MP).

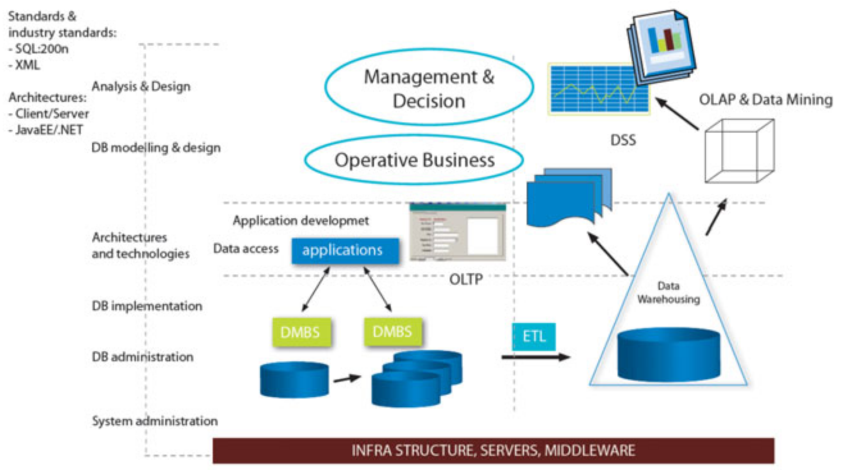


Fig. 1. The DBTechNet framework of subject areas and technologies

knowledge and skills for the success of the common cause, and (c) ready to work in order to bridge the gap between HE database technology teaching and the industrial practice, as well as that between the former and the corresponding vocational education and training (VET) practice.

Following the completion of the EU (Leonardo da Vinci) project DBTech Pro in 2005 [2], the second EU-funded DBTechNet project in currently progress: DBTech EXT (LLP Transversal Program, 01/2009 - 12/2010, [3]). The partnership has evolved to have its inner-circle (core) project participation include members from six (6) EU member states: seven (7) HE academic institutions, three (3) VET centres, and one (1) company. In addition, an outer-circle partnership has been established, its members currently representing nine (9) EU members states: thirteen (13) HE academic institutions, three (3) VET centres, and eight (8) I.T. companies/DBMS vendors. Outer-circle members act as project deliverables recipients and evaluators.

Two of of the main activities undertaken in DBTech Pro had to do with (a) the identification of the professional roles, the technologies, and the subject areas pertaining to today's requirements of the European database technology labour market, and (b) the compilation of a framework and a plan for the education and training content that comes to serve the requirements outlined in (a) [4,5].

Figure 1 summarises on the DBTechNet framework of database subject areas/technologies and, it is implicitly indicative of the professional roles involved.

2 The DBTech EXT Project

As stated in the introduction above, the project participation (both inner- and outer-circle type) involves HE institutions, VET centres, and I.T. companies from nine (9) EU member states. The two-year project has a budget of €508,391

(75% of which consists the contribution of the EU LLP Transversal Program), and it is due to conclude in December 2010.

Seven (7) work packages (WP) comprise the units of categorisation and organisation of the activities undertaken:

- WP1 involves the overall project co-ordination management
- WP2 focuses on the pedagogical value and the quality of the deliverables
- WP3 undertakes the compilation of the DBTechNet framework of courses, and their learning outcomes along the lines of the European Qualifications Framework (EQF) as well as the European Credit Transfer System (ECTS). On the basis of this framework, WP3 co-ordinates the development of the virtual laboratory workshops' (VLW) educational and training (E&T) content
- WP4 considers the vocational education and training (VET) dimension of the DBTechNet framework of courses and recommends the actions that need to be undertaken in order to have the E&T content comply with the requirements of the VET and the lifelong learning/training type course curricula
- WP5 conducts a survey on and reports on the current needs for post-graduate (i.e. MSc and PhD) studies on database technologies with emphasis on the professional dimension of the latter, considering the present and the near future needs of the European labour market
- WP6 and WP7 co-ordinate activities of dissemination and exploitation (respectively) of the project deliverables

Figure 2 summarizes on the DBTech EXT work packages and their interrelationships.

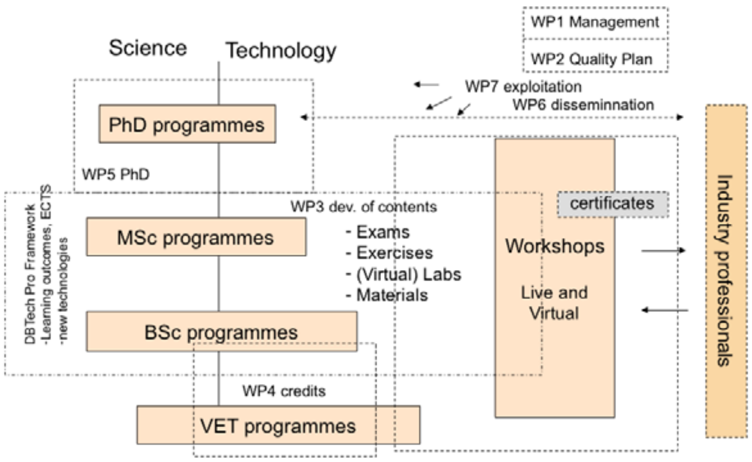


Fig. 2. The DBTech EXT work packages

3 Virtual Laboratory Workshops (VLWs)

Virtual laboratory workshops admittedly comprise the DBTech EXT project deliverable that has the potential to evolve further in the direction of facilitating the co-operation and the productive sharing of database technologies expertise across the European Union (EU) Higher Education (HE) institutions, the Vocational Education and Training (VET) and Lifelong Learning (LLL) centres on one hand, and the EU information technology (IT) industry on the other. The task is not an easy one to achieve. The collaboration framework can be successful only if it is laid on a sound foundation of mutual benefit: knowledge and training skills providers need invaluable feedback from the industry to make sure that the E&T content they develop and offer serves the present as well as the foreseen near future needs of the labour market, and the industry can highly benefit by having access to vendor-neutral, high quality E&T content on modern trends and practices in database technologies.

The seven (7) VLWs developed along the lines of the DBTech EXT project have as follows:

- Database Modeling and Semantics
- Data Access Patterns, Technologies and Frameworks
- Distributed, Replicated, Mobile, and Embedded Databases
- Concurrency Control and Recovery Technologies
- OLAP and Data Warehouses
- Business Intelligence and Knowledge Discovery from Databases
- XML and Databases

The E&T content of a typical DBTechNet VLW consists of:

- Tutorial slides
- Guidelines for self-study reading with the recommended bibliography
- Exercises with model answers
- Self-paced lab practicing material
- Hands-on laboratory exercises and case-studies
- Access to the appropriate software (commercial and/or open source)
- VLW participant's guide
- VLW teacher/trainer's guide
- Self-assessment multiple choice questionnaire (MCQ) for the VLW participants to identify the VLW topics which they need to improve their knowledge and skills on

One of the VLWs (Business Intelligence and Knowledge Discovery from Databases, [6]) includes a number of digital videos in its E&T content, as an act of probing one possible extension which is likely to be implemented next in all VLWs.

4 Internet-Based Teaching and Training in Practice

One of the most useful lessons learnt from conducting teaching and training sessions over the internet (in particular: when the E&T content includes hands-on training session as in the case of the DBTechNET VLWs) is that communication comprises the single-most decisive parameter for success.

The DBTechNet VLW E&T content is organised, alongside with all the DBTechNet material, in the DBTechNet portal [7]. The structure of the latter is such that each one VLW comprises a logical unit, within the portal environment. Each one VLW unit is assigned one or more portal user accounts who act(s) as the unit's (local) co-ordinator(s). The administrator of the portal assigns operational privileges to unit co-ordinators, enabling them to (a) administer the unit's (local) user membership, and (b) determine the accessibility level of each one item published within their unit, by choosing one from three possible levels: "public", "portal users", and "unit users, only" [8]. The unit co-ordinator(s) control the subset of the portal users population who are granted membership to the unit they co-ordinate, but they have no control over the administration of the portal users population; the latter requires portal administration authority. The portal is implemented by configuring the Moodle content management system to achieve the desirable functionality [9].

The information and communication technology (ICT) tools and services suite used for communicating with the VLW participants includes also the utilisation of the Alexander Technology Educational Institute (DBTech EXT inner-circle partner) video-on-demand server for organizing and managing the digital video content. For conducting synchronous tele-conferencing sessions involving the VLW instructor(s) and participants, use is made of the University of Malaga and the HAAGA-HELIA University of Applied Sciences (both: inner-circle DBTech EXT partners) Adobe Connect Pro web conferencing facilities [10].

Last but not least comes the software which is made available to the VLW participants in order to conduct their self-paced lab practicing and hands-on laboratory workshop sessions. To begin with, each VLW may utilize two types of software: open source/free or commercial. In both cases, care is taken so that the VLW participant does not deal with technical issues (i.e. the installation and tuning of the software used), and make sure that s/he focuses on the main topic(s) of the workshop. In this respect, use is made of the operating system virtualisation technology. For the open source/free software, the approach is pretty straightforward: use is made of free virtualisation technology software (e.g. VMWare[11], MS PC Virtual[12], Oracle/Sun VirtualBox[13]) to create a virtual machine (VM) image of a standard free operating system, including the latest open source/free versions of the VLW software, and the corresponding DBTech EXT tutorial/self-study E&T content. DBTech VLW participants are directed to the DBTech portal to download their VM images which they may then use on their own PCs by means of the corresponding (free) players [14].

Except from open source/free software, DBTech VLWs utilise commercial software made available via academic licensing [15,16]. To make it possible for the industrial and VET center participants to have access to such "for educational

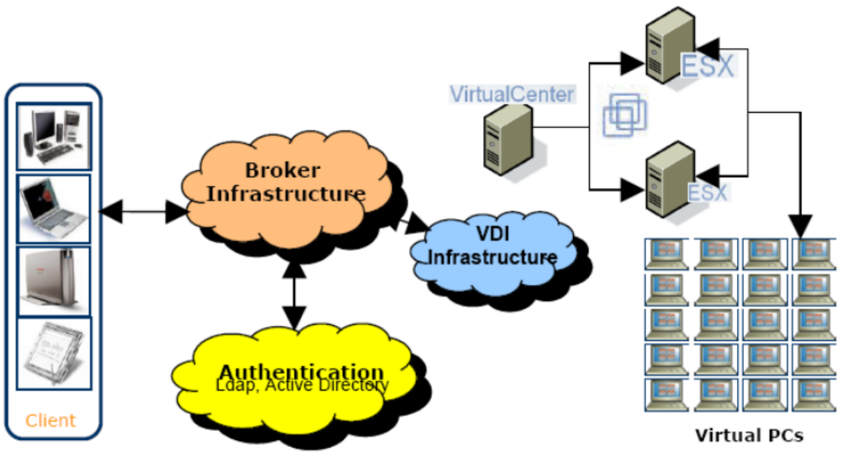


Fig. 3. The UMA Virtual Desktop Infrastructure Architecture

use, only" software resources, use is made of the University of Malaga (UMA) virtual desktop infrastructure and service (VDI) [17].

Figure 3 is taken from [18] and represents graphically the architecture of the UMA VDI infrastructure, consisting of:

- VMWare infrastructure virtualisation servers (Virtual centre plus multiple ESX VMWare servers),
- Connection broker,
- Re-compressors, and
- Additional servers: DNS servers, DHCP servers, databases, authentication servers (LDAP, ActiveDirectory).

The UMA PCVirtual service makes possible the creation of virtual computer images on the Virtual Desktop Infrastructure (VDI) server. All the VLW participant needs to access and use his/her VM image via the internet is a browser which supports the ActiveX technology (MS Internet Explorer), or the java applet technology (MS Internet Explorer, Mozilla Firefox, Safari, etc.) [18]. It is via the UMA PCVirtual service that VET center and non-academic/non-HE student participants of the OLAP and Data Warehousing, and Business Intelligence and Knowledge Discovery from Databases VLWs establish access to and use proprietary software like Microsoft SQL Server, and IBM Intelligent Miner for Data.

5 Conclusion

In this presentation, the whole range of the DBTech EXT project deliverables is presented and the audience are given the opportunity to realise the benefits of a 'no commitment/no risk' contact and collaboration with the DBTech EXT partnership. More specifically, the presentation includes a sample of the project's virtual laboratory workshops (VLWs) educational and training (E&T) content. The latter is freely available to academics and trainers who wish to incorporate it into their HE, and/or VET course curricula, as well as to I.T. professionals and companies who wish to utilise it in lifelong training sessions. The E&T content in question is subject to DBTech EXT internal, as well as to external evaluation with regard to its pedagogical value, and quality. In addition to the hands-on/internet-based VLWs on selected DB Technology topics, project deliverables include a VET provision plan for a framework of intermediate DBMS professional knowledge and skills course syllabi, plus an investigation of (a) the currently existing opportunities for professional certification of DB Technology knowledge and skills in the EU, and (b) the possibilities of opting for an advanced HE degree study (MSc and PhD) focusing on the stated professional role.

References

1. DBTechNet initiative, http://www.dbtechnet.org/
2. DBTech Pro project, http://myy.haaga-helia.fi/%7Edbms/dbtechnet/project2002-05_en.HTML
3. DBTech EXT project, http://dbtech.uom.gr/mod/resource/view.php?id=181
4. Connolly, T.M.: DBTech Pro Specification of Knowledge Areas (2005), http://myy.haaga-helia.fi/%7Edbms/dbtechnet/finalr/WP2
5. Dervos, D.A.: DBTech Pro Content Planning (2005), http://myy.haaga-helia.fi/%7Edbms/dbtechnet/finalr/WP4.PDF
6. Dervos, D.A., Evangelidis, G., Karamitopoulos, L., Aunimo, L., Aldana Montes, J.F., Farfán-Leiva, J.J., Siakas, K., Valkama, H.: DBTech EXT Virtual Laboratory Workshop: Business Intelligence. In: eRA-4 International Conference on the Contribution of I.T. to Science, Technology, Society and Education, Spetses, Greece (2009), http://dbtech.uom.gr/mod/resource/view.php?id=143
7. DBTechNet portal, http://dbtech.uom.gr/
8. Evangelidis, G., Pitsiougkas, E., Dervos, D.A., Laiho, M., Laux, F., Aldana-Montes, J.F.: DBTech portal: A Gateway to Education and Training for the European Database Technology Professional. In: eRA-4 International Conference on the Contribution of I.T. to Science, Technology, Society and Education, Spetses, Greece (2009), http://dbtech.uom.gr/mod/resource/view.php?id=144
9. Moodle CMS, a free web application, http://moodle.org/
10. Adobe Connect Pro, http://www.adobe.com/products/acrobatconnectpro/
11. VMWare, http://www.vmware.com/
12. MS PCVirtual, http://www.microsoft.com/windows/virtual-pc/
13. Oracle/VirtualBox, http://www.virtualbox.org/
14. Aunimo, L., Laiho, M., Lassila, A.: DBTech Database Virtual Laboratory Workshops in Teaching. In: 14th Workshop of the Special Interest Group on Experimental Interactive Learning in Industrial Management of the IFIP Working Group 5.7: "Experimental Learning on Sustainable Management, Economics and Industrial Engineering", Poliscript Politecnico di Milano, Italy (2010), ISBN 97888-6493-004-6

15. IBM academic initiative program,
http://www.ibm.com/developerworks/university/academicinitiative/
16. Microsoft MSDN Academic Alliance program,
http://msdn.microsoft.com/en-us/academic/
17. University of Malaga PCVirtual/VDI service, https://pcvirtual.cv.uma.es
18. Delgado, I.N., Roldán-García, M., Farfán-Leiva, J.J., Martti, L., Laux, F., Dervos, D.A., Aldana Montes, J.F.: Innovative Unity in the Diversity of Information Management Skills Teaching and Training across Europe. IADAT Journal of Advanced Technology on Education (IJAT-e) 3(3), 439–443 (2009)

New Frontiers in Business Intelligence: Distribution and Personalization

Stefano Rizzi*

DEIS - University of Bologna, V.le Risorgimento 2, 40136 Bologna, Italy

Abstract. To meet the new, more sophisticated needs of decision makers, a new generation of BI systems is emerging. In this paper we focus on two enabling technologies for this new generation, namely distribution and personalization. In particular, to support complex business scenarios where multiple partner companies cooperate towards a common goal, we outline a distributed architecture based on a network of collaborative, autonomous, and heterogeneous peers, each offering monitoring and decision support functionalities to the other peers. Then we discuss some issues related to OLAP query reformulation on peers, showing how it can be achieved using semantic mappings between the local multidimensional schemata of peers. Finally, as to personalization, we discuss the benefits of annotating OLAP queries with preferences, focusing in particular on how preferences enable peer heterogeneity in a distributed context to be overcome.

Keywords: Business Intelligence, Distributed Data Warehousing, User Preferences.

1 Introduction

Business intelligence (BI) transformed the role of computer science in companies from a technology for passively storing data into a discipline for timely detecting key business factors and effectively solving strategic decisional problems. However, in the current changeable and unpredictable market scenarios, the needs of decision makers are rapidly evolving as well. To meet the new, more sophisticated user needs, a new generation of BI systems (often labeled as *BI 2.0*) has been emerging during the last few years. Among the characterizing trends of these systems, we mention BI as a service, real-time BI, collaborative BI, and pervasive BI.

In this paper we focus on two enabling technologies for BI 2.0, namely distribution and personalization.

* Part of this work has been jointly carried out in collaboration with Matteo Golfarelli, Wilma Penzo, Elisa Turricchia (Univ. of Bologna, Italy), and Federica Mandreoli (Univ. of Modena and Reggio Emilia, Italy).

1.1 Motivating Scenario and Envisioned Architecture

Cooperation is seen today by companies as one of the major means for increasing flexibility and innovating so as to survive in today uncertain and changing market. Companies need strategic information about the outer world, for instance about trading partners and related business areas. Indeed, it is estimated that above 80% of waste in inter-company and supply-chain processes is due to a lack of communication between the companies involved.

In such a complex and distributed business scenario, where multiple partner companies/organizations cooperate towards a common goal, traditional BI systems are no longer sufficient to maximize the effectiveness of monitoring and decision making processes. Two new significant requirements arise:

- Cross-organization monitoring and decision making: Accessing local information is no more enough, users need to transparently and uniformly access information scattered across several heterogeneous BI platforms [8].
- Pervasive and personalized access to information: Users require that information can be easily and timely accessed through devices with different computation and visualization capabilities, and with sophisticated and customizable presentations [14].

The architecture we envision to cope with this scenario is that of a dynamic, collaborative network of peers, each hosting a local, autonomous BI platform (see Figure 1). Each peer relies on a local multidimensional schema that represents the peer's view of the business and offers monitoring and decision support functionalities to the network users. Users transparently access business information distributed over the network in a pervasive and personalized fashion. Access is secure, depending on the access control and privacy policies adopted by each peer.

A typical user interaction in this context is the following. A user formulates an OLAP query q by accessing the local multidimensional schema of her peer, p. She

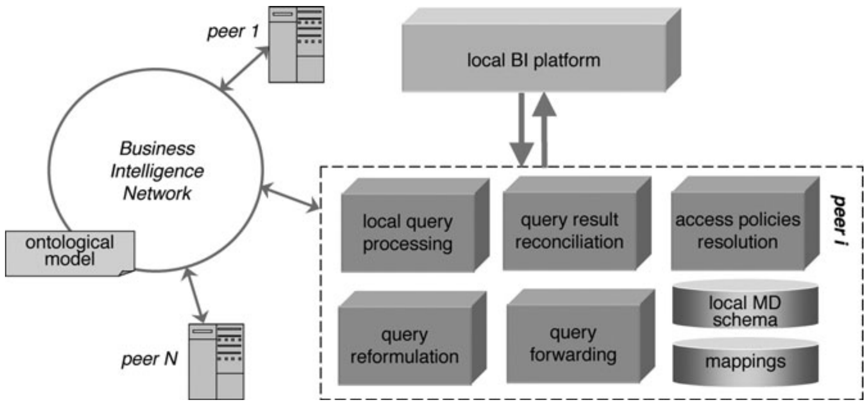


Fig. 1. Envisioned architecture for a collaborative BI network

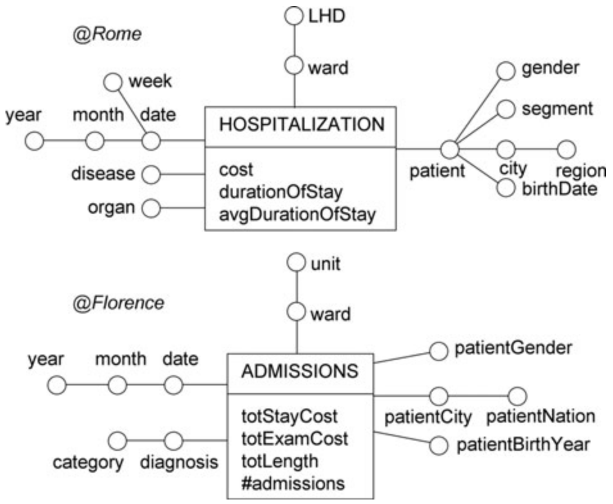


Fig. 2. Multidimensional schemata of related facts at two peers

can annotate q by a preference that enables her to rank the returned information according to her specific interests. To enhance the decision making process, q is forwarded to the network and reformulated on the other peers in terms of their own multidimensional schemata. Each involved peer locally processes the reformulated query and returns its (possibly partial or approximate) results to p. Finally, the results are integrated, ranked according to the preference expressed by the user, and returned to the user based on the lexicon used to formulate q.

In the working example we adopt in this paper, a set of local health-care departments participate in a collaborative network to integrate their data about admissions so as to enable more effective analysis of epidemics and health-care costs by the Ministry. For simplicity we will focus on two peers: the first, located in Rome, hosting data on hospitalizations at patient-level detail; the second, located in Florence, hosting data on admissions grouped by patient gender, residence city, and birth year. The underlying multidimensional schemata for these two peers are shown in Figure 2, using the Dimensional Fact Model notation [4].

2 Distribution

Although the integration of heterogeneous databases has been widely discussed in the literature, only a few works are specifically focused on strategies for data warehouse integration [2,16] and federation [1]. Indeed, in this context, problems related to data heterogeneity are usually solved by ETL (Extraction, Transformation, and Loading) processes that read data from several data sources and load them in a single multidimensional repository to be accessed by users. While this centralized architecture may fit the needs of old-style, stand-alone companies, it is hardly feasible in the context of a collaborative BI network, where the dynamic nature of the business, together with the independence and autonomy of peers, call for more sophisticated solutions.

Peer Data Management Systems (PDMSs [7]) have been proposed in the literature as architectures to support sharing of operational data across networks of peers while guaranteeing peer autonomy, based on interlinked semantic mappings that mediate between the heterogeneous schemata exposed by peers [10]. The architecture we outlined in the previous section is in line with the PDMS infrastructure, but requires a number of specific issues —mostly related to the multidimensional nature of the information exchanged— to be faced:

– Query reformulation on peers is a challenging task due to the presence of aggregation and to the possibility of having information represented at different granularities and under different perspectives in each peer.
– The strategic nature of the exchanged information and its multidimensional structure, as well as the presence of participants that belong to different organizations, require advanced approaches for security, ranging from proper access policies to data sharing policies that depend on the degree of trust between participants, as well as techniques for protecting against undesired information inference.
– Mechanisms for controlling data provenance and quality in order to provide users with information they can rely on should be provided. A mechanism for data lineage is also necessary to help users understand the semantics of the retrieved data and how these data have been transformed to handle heterogeneity.
– A unified, integrated vision of the heterogeneous information collected must be returned to users. To this end, object fusion functionalities that take into account the peculiarities of multidimensional data must be adopted.

As a first step in this direction, we are currently working on query reformulation. In particular, we devised a language for the definition of semantic mappings between the multidimensional schemata of peers, and we introduced a query reformulation framework that relies on the translation of these mappings towards a ROLAP platform. A basic multidimensional model is considered, where a fact (e.g., patient admissions) is associated to a set of coordinates called dimensions (e.g., ward and admission date) and is quantified by a set of numerical measures (e.g., the total cost of the stay). A dimension can be further described by a set of hierarchically-structured attributes connected by many-to-one associations (e.g., a patient lives in one city, that in turn belongs to one region). As to the workload, we consider OLAP queries that can be expressed in GPSJ (Generalized Projection - Selection - Join) form [6]:

$$\pi_{G,AGG(m)}(\sigma_P(\chi))$$

where π denotes a generalized projection, i.e., an aggregation of measure m using aggregate function $AGG()$ over the attributes in G; σ_P is a selection based on Boolean predicate P; and χ denotes the star join between the fact table and the dimension tables. For instance, the following query expressed at the Rome peer computes the total hospitalization cost of female patients for each region and year:

$$\pi_{region,year,SUM(cost)}(\sigma_{gender='F'}(\chi_{Rome}))$$

Now, let p and q be two peers in a collaborative BI network. The language we propose to express how the local multidimensional schemata of p maps onto the one of q includes five mapping predicates, namely same, equi-level, roll-up, drill-down, and related. Each mapping establishes a semantic relationship from a list of concepts (measures or attributes) of p (on the left side of the mapping predicate) to a list of concepts of q (on the right side of the mapping predicate), and enables a query formulated on p to be (exactly or approximately) reformulated on q. Optionally, a mapping can be associated with an encoding function that specifies how values of the left-side list of concepts can be obtained from values of the right-side list of concepts. If this function is available, it is used during query reformulation and data integration to return more query-compliant results to users.

While discussing in detail the whole set of mapping predicates and the query reformulation framework is outside the scope of this paper, we will give an intuition of the underlying mechanism with a basic example.

Example 1. The same predicate is used to state that a set c of measures in p has the same semantics than a set d of measures in q. If knowledge is available about how values of c can be derived from values of d, it can be expressed by two encoding functions $f : dom(c) \rightarrow \mathbb{R}$ and $g : dom(d) \rightarrow \mathbb{R}$. The semantics of these functions is that, whenever $f(c)$ is asked in a query at p, it can safely be rewritten as $g(d)$ at q. For instance,

$$< \mathsf{Rome.cost} > \mathsf{same}_{f,g} < \mathsf{Florence.totStayCost}, \mathsf{Florence.totExamCost} >$$

with

$$\begin{aligned} f(< \mathsf{cost} >) &= \mathsf{cost} \\ g(< \mathsf{totStayCost}, \mathsf{totExamCost} >) &= \mathsf{totStayCost} + \mathsf{totExamCost} \end{aligned}$$

states that measure cost can be derived by summing totStayCost and totExamCost.

Similarly, the roll-up predicate states that a set c of attributes in p is a roll-up of (i.e., it aggregates) a set d of attributes in q. If knowledge is available about how to roll-up values of d to values of c, it can be expressed by a non-injective encoding function $f : dom(d) \rightarrow dom(c)$ that establishes a one-to-many relation between values of c and values of d, and is used to aggregate data returned by q and integrate them with data returned by p. For instance,

$$< \mathsf{Rome.week} > \mathsf{roll\text{-}up}_f < \mathsf{Florence.date} >$$

with $f(< \mathsf{date} >) =< \mathsf{week} : weekOf(\mathsf{date}) >$, states that weeks are an aggregation of dates.

Now consider the OLAP query (formulated at Rome) asking for the weekly hospitalization costs:

$$\pi_{\mathsf{week},SUM(\mathsf{cost})}(\chi_{Rome})$$

This query group-by is reformulated using the roll-up mapping from week to date, while measure cost is derived using the same mapping:

$$\pi_{weekOf(\mathsf{date}),SUM(\mathsf{totStayCost}+\mathsf{totExamCost})}(\chi_{Florence})$$

□

3 Personalization

Personalizing e-services by allowing users to express preferences is becoming more and more common. When querying, expressing preferences is seen as a natural way to avoid empty results on the one hand, information flooding on the other. Besides, preferences allow for ranking query results so that the user may first see the data that better match her tastes.

Though a lot of research has been carried out during the last few years on database preferences (e.g., [11,3]), the problem of developing a theory of preferences for multidimensional data has been mostly neglected so far with a few exceptions [15,9,17,12]. Indeed, expressing preferences could be valuable in this domain because:

- Preferences enable users to focus on the most interesting data. This is particularly beneficial in the OLAP context, since multidimensional databases store huge amounts of data. Besides, OLAP queries may easily return huge volumes of data (if their group-by sets are too fine), but they may return little or no information as well. The data ranking entailed by preferences solves both these problems.
- During an OLAP session, the user often does not exactly know what she is looking for. The reasons behind a specific phenomenon or trend may be hidden, and finding those reasons by manually applying different combinations of OLAP operators may be very frustrating. Preferences enable users to specify a "soft" pattern that describes the type of information she is searching for.
- In a collaborative BI network like the one envisioned in Section 1.1, heterogeneity in peers' multidimensional schemata and data may lead to obtaining empty results when reformulating queries, while a query annotated with a preference can produce meaningful results even when a common schema is not defined and the searched data are not found.

The last motivation plays a key role in the distributed framework outlined in this work. In our health-care example, consider a query asking in Rome for some statistics on hospitalizations, aggregated at the finest level along the patient hierarchy. If the patient group-by were expressed as a "hard" constraint, no data from Florence could be returned because the patient granularity is not present there (see Figure 2). If the patient group-by is expressed in a "soft" form using a preference, instead, data aggregated by patient gender, city, and birth year can be returned from Florence. Though this granularity does not exactly match the one preferred by the user, the resulting information could be valuable in improving decision-making effectiveness. Similarly, a query in Rome could ask for data on hospitalizations whose duration of stay exceeds, say, 30 days. If the Florence peer stores no admissions yielding durations higher than 30, it can still return its admissions ranked by decreasing durations.

In [5] we presented myOLAP, an approach for expressing and evaluating OLAP preferences. From the expressiveness point of view, the main features of our approach can be summarized as follows:

– Preferences can be expressed not only on attributes, that have categorical domains, but also on measures, that have numerical domains. Remarkably, the preference constructors that operate on attributes take the presence of hierarchies into account.
– Preferences can also be formulated on the aggregation level of data, i.e., on group-by sets, which comes down to expressing preferences on schema rather than on instances.
– Preferences can be freely composed using the Pareto and prioritization operators, thus forming an algebra that can easily be incorporated into a multidimensional query language like MDX [13].

We close this section by showing how the two preference query suggested above can be formulated in myOLAP.

Example 2. The first query, asking for total hospitalization cost preferably aggregated by patient, can be annotated with the simple myOLAP preference FINEST(PATIENT), stating that data should be preferably aggregated at the finest group-by set along the PATIENT hierarchy. While at Rome the finest group-by is patient, at Florence the finest group-by is patientCity, patientGender, patientBirthYear. Of course, proper visualization techniques will be required for displaying results at different granularities together while preserving the typical intuitiveness and navigability of OLAP interfaces.

The second query asks for hospitalizations whose duration of stay exceeds 30 days. To express the constraint about duration in a "soft" way, a myOLAP preference like BETWEEN(durationOfStay,30,999) can be used. The best match for this preference are the hospitalizations whose duration of stay is higher than 30; however, if no such data are found, the returned hospitalizations are ranked in decreasing order by their durationOfStay values. □

4 Conclusions

In this paper we have outlined a peer-to-peer architecture for supporting distributed and collaborative decision-making scenarios. We have shown, using some examples, how an OLAP query formulated on one peer can be reformulated on a different peer, based on a set of inter-peer semantic mappings. The we have discussed the role of OLAP preferences in overcoming peer heterogeneity in both schemata and data.

Currently, we are finalizing a reformulation algorithm that relies on the translation of semantic mappings towards a ROLAP implementation. However, a large number of issues still need to be faced, as mentioned in Section 2. Our future work in this direction will be mainly focused on multidimensional-aware object fusion techniques for integrating data returned by different peers, on smart algorithms for routing queries to the most "promising" peers in the BI network, and on optimizing OLAP preferences in a distributed context.

References

1. Abiteboul, S.: Managing an XML warehouse in a P2P context. In: Eder, J., Missikoff, M. (eds.) CAiSE 2003. LNCS, vol. 2681, pp. 4–13. Springer, Heidelberg (2003)
2. Banek, M., Vrdoljak, B., Tjoa, A.M., Skocir, Z.: Automated integration of heterogeneous data warehouse schemas. IJDWM 4(4), 1–21 (2008)
3. Chomicki, J.: Preference formulas in relational queries. ACM TODS 28(4), 427–466 (2003)
4. Golfarelli, M., Rizzi, S.: Data Warehouse design: Modern principles and methodologies. McGraw-Hill, New York (2009)
5. Golfarelli, M., Rizzi, S., Biondi, P.: MYOLAP: An approach to express and evaluate olap preferences. IEEE Trans. Knowl. Data Eng. (to appear 2010)
6. Gupta, A., Harinarayan, V., Quass, D.: Aggregate-query processing in data warehousing environments. In: Proc. VLDB, Zurich, Switzerland, pp. 358–369 (1995)
7. Halevy, A.Y., Ives, Z.G., Madhavan, J., Mork, P., Suciu, D., Tatarinov, I.: The Piazza peer data management system. IEEE Trans. Knowl. Data Eng. 16(7), 787–798 (2004)
8. Hoang, T.A.D., Nguyen, B.: State of the art and emerging rule-driven perspectives towards service-based business process interoperability. In: Proc. Int. Conf. on Computing and Communication Technologies, Danang City, Vietnam, pp. 1–4 (2009)
9. Jerbi, H., Ravat, F., Teste, O., Zurfluh, G.: Applying recommendation technology in OLAP systems. In: Proc. ICEIS, Milan, Italy, pp. 220–233 (2009)
10. Kehlenbeck, M., Breitner, M.H.: Ontology-based exchange and immediate application of business calculation definitions for online analytical processing. In: Pedersen, T.B., Mohania, M.K., Tjoa, A.M. (eds.) DaWaK 2009. LNCS, vol. 5691, pp. 298–311. Springer, Heidelberg (2009)
11. Kießling, W.: Foundations of preferences in database systems. In: Proc. VLDB, Hong Kong, China, pp. 311–322 (2002)
12. Koutrika, G., Ioannidis, Y.: Answering queries based on preference hierarchies. In: Proc. VLDB, Auckland, New Zealand (2008)
13. Microsoft: MDX reference (2009), http://msdn.microsoft.com/en-us/library/ms145506.aspx
14. Rizzi, S.: OLAP preferences: a research agenda. In: Proc. DOLAP, Lisbon, Portugal, pp. 99–100 (2007)
15. Stefanidis, K., Pitoura, E., Vassiliadis, P.: Adding context to preferences. In: Proc. ICDE, Istanbul, Turkey, pp. 846–855 (2007)
16. Torlone, R.: Two approaches to the integration of heterogeneous data warehouses. Distributed and Parallel Databases 23(1), 69–97 (2008)
17. Xin, D., Han, J.: P-cube: Answering preference queries in multi-dimensional space. In: Proc. ICDE, Cancún, México, pp. 1092–1100 (2008)

Effectively Monitoring RFID Based Systems

Fabrizio Angiulli[2] and Elio Masciari[1]

[1] ICAR-CNR – Institute of Italian National Research Council
masciari@icar.cnr.it
[2] DEIS-UNICAL
Via P. Bucci, 87036 Rende (CS) Italy
fangiulli@deis.unical.it

Abstract. Datastreams are potentially infinite sources of data that flow continuously while monitoring a physical phenomenon, like temperature levels or other kind of human activities, such as clickstreams, telephone call records, and so on. Radio Frequency Identification (RFID) technology has lead in recent years the generation of huge streams of data. Moreover, RFID based systems allow the effective management of items tagged by RFID tags, especially for supply chain management or objects tracking. In this paper we introduce SMART (Simple Monitoring enterprise Activities by RFID Tags) a system based on outlier template definition for detecting anomalies in RFID streams. We describe SMART features and its application on a real life scenario that shows the effectiveness of the proposed method for effective enterprise management.

1 Introduction

In this paper we will focus on Radio Frequency Identification (RFID) data streams monitoring as RFID based systems are emerging as key components in systems devoted to perform complex activities such as objects tracking and supply chain management. Sometimes RFID tags are referred to as electronic bar codes. Indeed, RFID tags use a signal that contains basic identification information about a product. Such tags can be used to track a product from manufacturer through distribution and then on to retailers. These features of RFID tags open new perspectives both for hardware and data management. In fact, RFID is going to create a lot of new data management needs. In more details, RFID applications will generate a lot of so called "thin" data, i.e. data pertaining to time and location. In addition to providing insight into shipment and other supply chain process efficiencies, such data provide valuable information for determining product seasonality and other trends resulting in key information for the companies management. Moreover, companies are exploring more advanced uses for RFIDs. For instance, tire manufacturers plan to embed RFID chips in tires to determine tires deterioration. Many pharmaceutical companies are embedding RFID chips in drug containers to better track and avert the theft of highly controlled drugs. Airlines are considering RFID-enabling key onboard parts and supplies to optimize aircraft maintenance and airport gate preparation turnaround time. Finally, US Department of Defense recognizes the value of expanding the global RFID infrastructure and propose a RFID-capable supply chain as a critical element of defense transformation.

Such a wide variety of systems for monitoring data streams could benefit of the definition of a suitable technique for detecting anomalies in the data flows being analyzed. As a motivating example you may think about a company that would like to monitor the mean time its goods stay on the aisles. The goods are tagged by an RFID tag so the reader continuously produces readings reporting the electronic product code of the item being scanned, its location and timestamp, this information can be used for signaling that the item lays too much on the shelf since it is repeatedly scanned in the same position. It could be the case that the package is damaged and consequently customers tend to avoid the purchase. If an item exhibits such a feature it deserves further investigation. Such a problem is relevant to a so huge number of application scenario that make impossible to define an absolute notion of anomalies (in the follow we refer to anomalies as outliers). In this paper we propose a framework for dealing with the outliers detection problem in massive datastreams generated in a network environment for objects tracking and management. The main idea is to provide users a simple but rather powerful framework for defining the notion of outliers for almost all the application scenarios at an higher level of abstraction, separating the specification of data being investigated from the specific outlier characterization.

Our approach is based on the definition of a template for specifying outlier queries on datastreams. The template need to be powerful enough to model all the interesting surveillance scenarios. In this respect, it should allow the definition of four components, namely: the reference population P (due to the infinite nature of datastreams we need to work on significant sample of the whole stream) depending on the application context, the kind of objects (O) to be monitored, the attributes (A) of the population used for signing out anomalies, the outlier definition by means of a suitable function $\mathcal{F}(P, A, O) \rightarrow \{0, 1\}$. Finally, a mapping function that for a given template and data stream schema, resolves the template in a set of "outlier continuous queries" to be issued on the nodes generating the datastreams being monitored.

The function $\mathcal{F}$ to be used can be any stream oriented implementation of an outlier detection technique and it is completely orthogonal to the template definition since our goal is to design a management system that could help the decision makers in the analysis of huge streams of data. In particular, we provide a modular environment that provides several features for managing the company streams. Consider again the problem of singling out items lying on the shelf too much time. In this case the population of reference (P) is composed by all the items of the same kind placed in the same shelf during the last year, the objects of interests (O) are items, the attribute of interest (A) is the time of stay on the shelf, while the outlying function ($\mathcal{F}$) could be a statistical test stating that an object becomes an outlier if its time of stay deviates too much from the mean time of stay of the population (e.g. is more than 3 times further the standard deviation from the mean). This kind of template is always active at the nodes being monitored and could be modified by the master according to the (possible) concept drift. As an example of concept drift consider the case that due to decreased sales due to financial crisis the average stay of an item increases of some minutes, this information must be kept into account by the master that should modify the templates accordingly.

2 Related Work

The problem of representing and querying data streams has been investigated extensively in recent years due to increasing number of applications that generate huge amounts of data that flow continuously over communication networks. In particular many works address the problem of producing (approximate) answers to queries issued on an (infinite) datastream. Concerning the general task of querying the physical world (wired or wireless sensors can be modeled nearly the same of RFID reader networks) the problem of representing and querying data streams has been recently investigated from different viewpoints. Many of the systems proposed focused on sensor network readings management, which could be considered a more general case of RFID data stream. In the following we will briefly recall both theoretical and applied results on these topics. In [5] the authors analyze the space requirements of algorithms that operate on data streams and thus make only one (or a small number of) pass(es) over input data. In particular the authors treat the problem from different viewpoint, one-pass vs. multi-pass algorithms, deterministic vs. randomized and exact vs. approximation algorithms. In [2] the problem of modeling infinite datastreams and answering to continuous queries is discussed. In particular a system architecture for data stream management is presented. A query processing mechanism called *Eddies* that continuously reorder operators in the query plan as it runs is described in [1]. In particular they introduce the notion of synchronization barriers and moments of symmetry. In [8] a new class of samples called *icicles* is described. Icicles are samples that adapt to changes in the workload based on the relevance of a tuple for query answering.

Systems for managing and querying data generated by sensor networks are generally classified according to the approach adopted to execute queries. In centralized systems the access to the sensor network is separated from query evaluation. That is, data are continuously extracted from sensors and stored into a database server. Thus, queries are issued on the centralized database. ALERT[7] is one of the most popular centralized systems, and is widely employed in the context of environmental data processing. In the distributed approach (which is adopted by COUGAR[6] and TinyDB[4] systems) data are extracted from sensors only if they must be accessed to evaluate some queries.

The problem of efficiently managing RFID data has been studied w.r.t. different application scenarios. Specifically, RFID data management it has been studied in [3], where the problem of defining an efficient warehousing model along with techniques for summarizing and indexing data has been investigated.

More in detail, in [3] the authors propose a model that eliminates the redundancy in RFID data. For instance, the proposed model collapse multiple RFID readings for the same tag coming from the same reader, by assigning the interval of stay $(time_{in}, time_{out})$ instead of every timestamps . To further compress RFID data they register a single transition for items that are packed together (e.g. CDs in the same pallet). Finally, they introduced a model based on hierarchy of summary called *RFID-Cuboids* of the RFID data aggregated at different abstraction levels in order to allow different kinds of data analysis. In [10] the author exploited a matrix of RFID tags in order to track data. More in details they collected trajectories followed by tagged objects in order to extract typical activity phenomenon (i.e. goods traveling the same path through the supply chain).

Finally, in the context of Data Stream Languages the ESL language [9] has been proposed to extend SQl-3 capabilities in order to deal with the peculiar features of datastreams. It provides a wide collection of operators for defining stream sources and continuous queries on them, In this paper we will refer mainly to ESL syntax for the implementation of our monitoring system.

3 Preliminaries

3.1 RFID Data Streams

An RFID system consists of three components: the *tag*, the *reader* and the *application* which uses RFID data. Tags consist of an antenna and a silicon chip encapsulated in glass or plastic. Tags for commercial use contain a very small amount of information. For instance, many kinds of tag contain only a code number (denoted as *Electronic Product Code* (*epc*)). Currently, 128-bit *epc* is the most prevailing version and contains information about the manufacturer, the category of the object and a serial number that identifies the specific object being monitored. Regarding the transmission mode, tags can be passive, active or semi-active. An active tag contains some type of power source on the tag, whereas the passive tags rely on the radio signal sent by the reader for power. Most RFID applications today use passive tags because they are much cheaper to manufacture. However, the lack of power poses significant restrictions on the tags ability to perform computations and communicate with the reader. It must be within range of the reader in order to work properly. Semi-active tags use a battery to run the microchips circuitry but not to communicate with the reader. RFID readers or receivers are composed of a radio frequency module, a control unit and an antenna to query electronic tags via radio frequency (RF) communication. They also include an interface that communicates with an application (e.g., the check-out counter in a store). Readers can be hand-held or mounted in specific locations in order to ensure they are able to read the tags as they pass through a *query zone* that is the area within which a reader can read the tag. The query zone are the locations that must be monitored for application purposes.

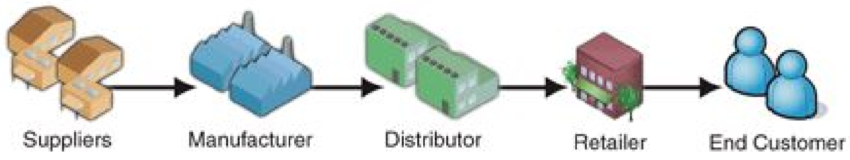


Fig. 1. Supply Chain Scenario

In order to explain the typical features of an RFID application we consider a supply chain such as the one depicted in Figure 1.

The chain from the farm to the customer has many stages. At each stage goods are typically delivered to the next stage, but in some case a stage can be missing. The following three cases may occur: 1) the goods lifecycle begin at a given point (i.e. production stages, goods are tagged there and then move through the chain) and thus the reader in the zone register only departures of goods, we refer to this reader as *source reader*; 2) goods are scanned by the reader both when they arrive and they leave the aisle, in this case we refer to this reader as *intermediate reader*; 3) goods are scanned and the tag is killed, we refer to this reader as *destination reader*.

RFID data can be exploited by traditional operational data management systems in order to perform operational tasks such as good delivery, packaging etc... and by data analysis applications used to support managers in making strategic level decisions. In the next section we consider the latter type of applications. In such applications we have to deal with the redundant nature of RFID data that being continuously scanned generate multiple readings for the same object thus making the analysis task quite intriguing.

A RFID stream is (basically) composed of an ordered set of n sources (i.e., tag readers) located at different positions, denoted by $\{r_1, \ldots, r_n\}$ producing n independent streams of data, representing tag readings. Each RFID stream can be basically viewed as a sequence of triplets $\langle id_r, epc, \tau_s \rangle$, where:

1. $id_r \in \{1, .., n\}$ is the tag reader identifier (observe that it implicitly carries information about the spatial location of the reader) ;
2. epc is the product code read by the source identified by id_r;
3. τ_s is a *timestamp*, i.e., a value that indicates the time when the reading epc was produced by the source id_r.

This simple schema could be enriched with additional information about locations and products as will be explained in section 4.

4 Statement of the Problem

In our model, *epc* is the identifier associated with a single unit being scanned (this may be a pallet or a single item, depending on the level of granularity chosen for tagging the goods being monitored).

This basic schema is simple enough to be used as a basic schema for a data stream environment, anyway since more information are needed about the outlier being detected we can access additional information by using some auxiliary tables maintained at a $Master$ site as shown in figure 2. More in detail, the $Master$ maintains an intermediate local warehouse of RFID data that stores information about items, items' movements, product categories and locations and is exploited to provide details about RFID data upon user requests. The information about items' movements are stored in the relation $ItemMovement$ and the information about product categories and locations are stored in the relations *Product* and *Locations*, respectively. These relations represent, respectively, the *Product* and the *Location* hierarchy. Relation *EPCProducts* maintains the association between *epcs* and product category, that is, every *epc* is associated to a tuple at the most specific level of the *Product* hierarchy. Finally, RFID readers constitute the most specific level of the *Location* hierarchy.

More in detail, the main characteristics of the *Product* and the *Location* hierarchy can be described as follows:

– $ProductCategories$: this hierarchy takes into account, using different granularity levels, the differences among products. The bottom of the hierarchy groups items having common features, while the intermediate nodes represent the different aggregation that could be considered (e.g., *bread* $\rightarrow$ *bakey* $\rightarrow$ *perishable* $\rightarrow$ *foodstuffs*);
– *Locations*: this hierarchy has as bottom the single readers, and as intermediate nodes regions sharing the same goal (e.g., *reader* $\rightarrow$ *aisle* $\rightarrow$ *warehouse* $\rightarrow$ *docks* $\rightarrow$ *city*).

This information can be accessed when user requires aggregate information about readings. $ItemMovements$ contains tuples of the form $\langle epc, DL\rangle$, where *epc* has the usual meaning, and DL is string built as follows: each time an *epc* is read for the first time at a node N_i a trigger fires and DL is updated appending the node identifier.

In the following we define a framework for integrating Data Stream Management Systems (DSMS) technologies and outlier detection framework in order to effectively manage outliers in RFID datastreams. In particular we will exploit the following features:

– The definition of a template for specifying outlier queries on datastreams that could be implemented on top of a DSMS by mapping the template in a suitable set of continuous queries expressed in a continuous query language language such as ESL[9];
– The template need to be powerful enough to model all the interesting surveillance scenarios. In this respect, it should allow the definition of four components, namely:
  1. the kind of objects (O) to be monitored,
  2. the reference population P (due to the infinite nature of datastream) depending on the application context,

3. the attributes (A) of the population used for signing out anomalies,
4. the outlier definition by means of a suitable function $\mathcal{F}(P, A, O) \rightarrow \{0, 1\}$.

– A mapping function that for a given template and DSMS schema, resolve the template in a set of outlier continuous queries to be issued on the datastream being monitored.

The basic intuition behind the template definition is that we want to run an aggregate function that is raised by the *Master* and then instantiated on a subset of nodes in the network as shown in figure 2, it is easy to see that the system works in a master-slave way. An incoming stream is processed at each *node* where the template is activated by the $Master$ that issued the request for monitoring the stream. Once a possible outlier is detected, it is signaled to the $Master$. The $Master$ maintains management information about the network and some additional information about the items using two auxiliary tables $OutlierMovement$ and $NewTrend$. In the $OutlierMovement$ table it stores information about the outlying objects, in particular it stores their identifiers and the paths traveled so far as explained above for $ItemMovements$. The $NewTrend$ table stores information about objects that are not outliers but instead they represent a new phenomenon in the data. It contains tuples of the form $\langle epc, N, \tau_a, \tau_l, \rangle$, where N is a node, τ_a and τ_l are, respectively, the arrival time and the latency time (i.e. the time interval spent at node N by the *epc*). The latter table is really important since it is intended to deal with the concept drift that could affect the data. In particular, when items are marked as unusual but they are not outliers (as in the case of varied selling rates) they are recorded for later use since they are the symptom of a new trend for items being monitored . Once the new trend has been consolidated, new statistics for the node where the objects appeared will be computed at $Master$ level and then forwarded to the pertaining node in order to update the parameters of its population.

As mentioned above candidate outliers are signaled at node level and they are managed by the $Master$. More in detail, as a possible outlier is signaled by a given node the $Master$ stores it in the $OutlierMovement$ table along with its path if it is recognized as an anomaly or in the $NewTrend$ table if a signaled item could represent the symptom of a new trend in data. To summarize, given a signaled object o two cases may occur: 1) o is an outlier and then it is stored in the *Outlier* table; 2) o represents a new trend in data distribution and then it should not be considered as an outlier and we store it in the $NewTrend$ table. To better understand such a problem we define three possible scenarios on a toy example.

Example 1. Consider a container (whose *epc* is p_1) containing dangerous material that has to be delivered through check points c_1, c_2, c_3 in the given order and consider the following sequence of readings: $Seq_A = \{(p_1, c_1, 1), (p_1, c_1, 2), (p_1, c_2, 3), (p_1, c_2, 4), (p_1, c_2, 5), (p_1, c_2, 6), (p_1, c_2, 7), (p_1, c_2, 8), (p_1, c_2, 9), (p_1, c_2, 10), (p_1, c_2, 11), (p_1, c_2, 12)\}$. Sequence A correspond to the case in which the pallet tag is read repeatedly at the check point c_2. This sequence may occur because: i) the pallet (or the content) is damaged so it can no more be shipped until some

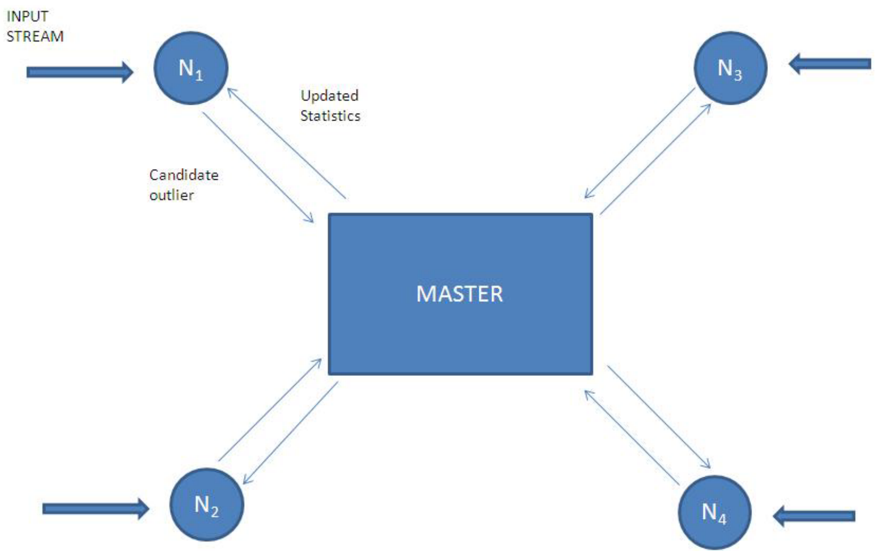


Fig. 2. Structure of our RFID stream environment

recovery operation has been performed, ii) the shipment has been delayed. Depending on which one is the correct interpretation different recovery action need to be performed. To take into account this problem in our prototype implementation we maintain appropriate statistics on latency time at each node for signaling the possible outlier. Once the object has been forwarded to the $Master$ a second check is performed in order to store it either in $OutlierMovement$ or in $NewTrend$ table. In particular, it could happen that due to new shipping policy additional checks have to be performed on dangerous material, obviously this will cause a delay in shipping operations, thus the tuple has to be stored in the $NewTrend$ table.

Consider now a different sequence of readings: $Seq_B = \{(p_1, c_1, 1), (p_1, c_1, 2), (p_1, c_1, 3), (p_1, c_1, 4), (p_1, c_3, 5), (p_1, c_3, 6), (p_1, c_3, 7), (p_1, c_3, 8), (p_1, c_3, 9), (p_1, c_3, 10), (p_1, c_3, 11), (p_1, c_3, 12)\}$. Sequence B correspond to a more interesting scenario, in particular it is the case that the pallet tag is read at check point c_1, is not read at check point c_2 but is read at checkpoint c_3. Again two main explanation could be considered: i) the original routing has been changed for shipment improvement, ii) someone changed the route for fraudulent reason (e.g. in order to steal the content or to modify it). In this case suppose that the shipping plan has not been changed, this means that we are dealing with an outlier then we store it in the $OutlierMovement$ table along with its path.

Finally, consider the following sequence of readings regarding products p_1, p_2, p_3 that are frozen foods, and product p_4 that is perishables, all readings being generated at a freezer warehouse c: $Seq_C = \{(p_1, c, 1), (p_2, c, 2), (p_3, c, 3), (p_4, c, 4), (p_1, c, 5), (p_2, c, 6), (p_3, c, 7), (p_4, c, 8), (p_1, c, 9), (p_2, c, 10), (p_3, c, 11), (p_4, c, 12)\}$. Obviously, p_4 is an outlier for that node of the supply chain and this can be easily recognized using a distance based outlier function since its expiry date greatly deviates from the expiry dates of other goods.

4.1 Template Definition and Functionalities

The Template in a Short. In this section we will describe the functionalities and syntax of the $Template$ introduced so far. A $Template$ is an aggregate function that takes as input a stream. Since the physical stream could contain several attributes as explained in previous sections we allow *selection* and *projection* operation on the physical stream. As will be clear in next section we will use a syntax similar to ESL with some specific additional features pertaining to our application scenario. This filtering step is intended for feeding the reference population P. In particular, as an object is selected at a given node it is included in the reference population for that node using an $Initialize$ operation, it persists in the reference population as a *Remove* operation is invoked (it can be seen as an $Initialize$ operation on the tuples exiting the node being monitored).

We recall that a RFID tagged object is scanned multiple times at a given node N so when the reader no more detects the RFID tag no reading is generated. First time an object is read a $Validate$ trigger fires and send the information to the $Master$ that eventually updates the $ItemMovement$ table. In response to a $Validate$ trigger the $Master$ performs a check on the item path, in particular it checks if shipping constraints are so far met. In particular, it checks the incoming reading for testing if the actual path so far traveled by current item is correct. This check can be performed by the following operations: 1) selection of the path for that kind of item stored in $ItemMovement$, 2) add the current node to the path, 3) check the actual path stored in an auxiliary table *DeliveryPlans* storing all the delivery plans (we refer to this check as *DELIVERY CHECK*). This step is crucial for signaling path anomalies since as explained in our toy examples that source of anomaly arise at this stage. If the item is not validated the $Master$ stores the item information in order to solve the conflict, in particular it could be the case that delivery plans are changing (we refer to this check as *NEW PATH CHECK*) so information is stored in $NewTrend$ table for future analysis , otherwise it is stored in the $OutlierMovement$ table. To better understand this behavior consider the Seq_B in example 1. When the item is first time detected at node c_3 the $Validate$ trigger fires, the path so far traveled for that object is retrieved obtaining $path = c_1$, the current node is added thus updating $path = c_1.c_3$ but when checked against the actual path stored in *DeliveryPlans* an anomaly is signaled since it was supposed to be $c_1.c_2.c_3$. In this case the item is stored in the $OutlierMovement$ table and the $Master$ signals for a recovery action. It works analogously for Seq_A as explained in example 1.

When an *epc* has been validated it is added to the reference population for that node (P_N) then it stays at the node and is continuously scanned. It may happen that during its stay at a given node an *epc* could not be read due to temporary electro magnetic field problem, we should distinguish this malfunction from the "normal" behavior that arise when an item is moved for shipping or (in case of destination nodes) because it has been sold. To deal with this feature we provide a trigger $Forget$ that fires when an object is not read for a (context depending) number of reading cycles (we refer in the following as *TIMESTAMP CHECK*. We point out that this operation is not lossy since at each node we maintain (updated) statistics on items. When $Forget$ trigger runs, it removes the "old" item from the actual population and update the node statistics. Node statistics (we refer hereafter to them as $model_N$ where N is the node they refer to) we take into account for outlier detection are: number of items grouped by product category ($count$), average time spent at the node by items belonging to a given category (m), variance for items (v) belonging to a given category, maximum time spent at the current node by items belonging to a given category (max_t), minimum time spent at the current node by items belonging to a given category (min_t). By means of the reference population P_N and the node statistics $model_N$ the chosen outlier function checks for anomalies. In particular, we can search for two kind of anomalies: 1) *item based* anomalies, i.e. anomaly regarding the item features, in this case we will run a distance-based outlier detection function; 2) *time based* anomalies, i.e. anomaly regarding arrival time or latency time, in this case we will run a statistical based outlier detection function.

In the following we report the tables structure:

$DeliveryPlans(id, actualPath, expectedTimestamp, maxTimestamp)$.
Wherein: id identifies the item being monitored, $actualPath$ is the scheduled path for the item, $expectedTimestamp$ is the suitable time value for an item to be delivered, $maxTimestamp$ is the maximum time value after that the item has to be signaled as an outlier;

$ItemMovement(id, path, timestamp)$. Wherein: id identifies the actual item, $path$ is the actual path traveled so far and $timestamp$ is the actual time stamp;

$Population(id, timestamp)$. The attributes have the same meaning above explained and the table is updated with those items exhibiting normal behaviors;

$OutlierMovement(id, path, timestamp)$. The attributes have the same meaning above explained and the table is updated with those items exhibiting exceptional behaviors;

$NewTrend(id, path, timestamp)$. The attributes have the same meaning above explained and the table is updated with those items that deviated from normal behavior but could not be considered as outliers, this table is intended to take into account the possible concept drift in data.

The above functionalities are summarized in the following table where we report for each action the expected results depending on the response to the check being performed and if it acts on the population (P) or the model (M) and the tables involved in the (eventual) updates:

4.2 The RFID-$\mathcal{T}$ Syntax

In this section we formalize the syntax for template definition. For basic stream operation we will refer to ESL syntax[9].

Cause	Action	Response	P	M	Updates
Validate Trigger	DELIVERY CHECK	OK	X	X	ItemMovemt
Validate Trigger	DELIVERY CHECK	FAIL			NewTrend or OutlierMovement
DELIVERY CHECK	NEW PATH CHECK	OK			NewTrend
DELIVERY CHECK	NEW PATH CHECK	FAIL			OutlierMovement
Forget Trigger	TIMESTAMP CHECK	OK	X	X	
Forget Trigger	TIMESTAMP CHECK	FAIL	X	X	

Fig. 3. Main action performed for items monitoring

CREATE STREAM	< *name* >
ORDER BY	< *attribute* >
SOURCE	< *systemnode* >
DEFINE OUTLIER TEMPLATE	< *name* >
ON STREAM	< *streamname* >
REFERENCE POPULATION	(< *definepopulation* >)
MONITORING	(< *target* >)
USING	< *outlierfunction* >
< *definepopulation* >	INSERT INTO < *PopulationName* >
	SELECT < *attributelist* >
	FROM < *streamname* >
	WHERE < *conditions* >
< *target* >	< *attributelist* > | < *aggegatefunction* >
< *outlierfunction* >	< *distancebased* > | < *statisticalbased* >

Fig. 4. Template Definition syntax

```
AGGREGATE           <Function Name> <Type>(Next Real) : Real
{ TABLE             <Table Name> (<attribute list>);
INITIALIZE:         { INSERT INTO <Table Name> VALUES (Next, 1); }
ITERATE:            { UPDATE <Population Name> SET <update condition>;
INSERT INTO RETURN  SELECT <output attribute> FROM <Table Name> }
TERMINATE :         {} }
```

Fig. 5. Aggregate Function syntax

The first step is to create the stream related to nodes being monitored. Once the streams are created at each node the *Template* definition has to be provided.

Aggregate function can be any *SQL* available function applied on the reference population as shown in Fig. 5, where *Return* and *Next* have the same interpretation as in *SQL* and < *Type* > can be any *SQL* aggregate function. An empty *TERMINATE* clause refer to a non-blocking version of the aggregate.

As the template has been defined it must be instantiated on the nodes being monitored. In particular triggers *Validate* and *Forget* are activated at each node. As mentioned above they will continuously update the reference population and node and *Master* statistics. The syntax of these triggers is shown in figure 6.

```
CREATE TRIGGER          Validate
BEFORE                  INSERT ON <Population Name>
REFERENCING NEW AS      NEW READING
IF PATH_CHECK > 0       INSERT INTO <Population Name> VALUES (NEW READING)
ELSE IF DELIVERY_CHECK  INSERT INTO NewTrend VALUES (NEW READING)
ELSE                    INSERT INTO OutlierMovement VALUES (NEW READING)
-----------------------------------------------------------------------------
CREATE TRIGGER          Forget
AFTER                   INSERT ON <Population Name>
REFERENCING OLD AS      OLD READING
IF TIMESTAMP_CHECK      {DELETE FROM <Population Name> OLD READING
                        UPDATE STATISTICS ON <Population Name> }
```

Fig. 6. Validate and Forget Trigger

```
CREATE FUNCTION        PATH_CHECK RETURN NUMBER AS
BEGIN
                SELECT COUNT(*)
                FROM   ItemMovement I, DeliveryPlans D
                WHERE  I.id=D.id AND
                       I.path=D.actualPath;
END
```

Fig. 7. PATH_CHECK syntax

We point out again that *Validate* trigger has the important side-effect of signaling *path* outliers. We point out that the above presented definition is completely flexible so if the user may need a different outlier definition she simply needs to add its definition as a plug-in in our system.

5 A Case Study

In this section, we present a real-life application scenario where we exploited our system for detecting outliers in RFID data streams. The streams are related to a set of pallets of tinned foods being tracked from the farm to distribution centers through all stages of the productive chain of a food company. We analyzed about 100 streams, belonging to 5 main product category (daily production): 1) `Tuna Fish`, whose readings are generated by 5000 tagged pallets storing 500 cans each; 2) `Tomato`, whose readings are generated by 6500 tagged pallets storing 40 cans each; 3) `Syrupy Peach`, whose readings are generated by 2000 tagged pallets storing 50 cans each; 4) `Meat`, whose readings are generated by 3500 tagged pallets storing 600 cans each and 5) `Ice Cream` whose readings are generated by 3000 tagged pallets storing 400 cans each.

The supply chain has 10 nodes, 3 nodes at production sites (*source readers* for RFID analysis purpose), 5 nodes at distribution centers (*intermediated readers*) and 2 selling points (*destination readers*).

```
CREATE FUNCTION       DELIVERY_CHECK RETURN NUMBER AS
DECLARE
                result  NUMBER
BEGIN
                SELECT −1INTO result
                FROM   ItemMovement I, DeliveryPlans D
                WHERE  I.id=D.id AND
                       I.timestamp>D.maxTimestamp;
UNION
                SELECT −1INTO result
                FROM   ItemMovement I, DeliveryPlans D
                WHERE  I.id=D.id AND
                       I.timestamp<D.expectedTimestamp;
UNION
                SELECT −1INTO result
                FROM   ItemMovement I, DeliveryPlans D
                WHERE  I.id=D.id AND
                       I.timestamp>D.expectedTimestamp AND
                       I.timestamp<D.maxTimestamp;
RETURN result
END
```

Fig. 8. DELIVERY_CHECK syntax

```
CREATE FUNCTION       TIMESTAMP_CHECK (forgetTime IN NUMBER) RETURN NUMBER AS
BEGIN
                SELECT *
                FROM   PopulationName P
                WHERE  P.timestamp+forgetTime < CURRENT_TIME;
END
```

Fig. 9. TIMESTAMP_CHECK syntax

We tested our prototype for one month monitoring the readings generated at each node in particular we focused on monitoring some parameters as requested by the company management: *delivery efficiency*, *packaging quality* and *weight management* control. While the first two parameters are clearly understandable the third parameter is quite interesting since tinned food passes through many stages from the raw materials collection to the final tinned products, during this working stages food has a natural weight decrease that is quite predictable, so the goal of the monitoring stage is to avoid theft during the production stages by workers that steal material trying to hide it as a natural weight loss.

In the table 1 we report the statistics generated at each node. In particular for each node we report the average population, the average stay time, the maximum stay time and minimum stay time, the *path* outlier detected at that node, and the *item* based outlier and the *time* based outlier. Note that node N_1 is the first production node so only *time* based outlier could be detected at that node.

Table 1. Results for SMART monitoring for a supply chain

node	*Population size*	*Stay Time*	*Max Stay Time*	*Min Stay Time*	*Path Outlier*	*Item Outlier*	*Time Outlier*
N_1	2000	40	49	28	0	0	4
N_2	1227	27	41	21	0	20	3
N_3	774	31	43	23	0	15	4
N_4	300	22	33	17	7	8	1
N_5	600	26	42	22	5	25	2
N_6	450	21	29	17	4	9	5
N_7	250	18	21	13	3	13	3
N_8	400	20	25	17	6	14	2
N_9	1100	34	39	26	8	22	1
$N_1$0	900	28	35	23	7	12	0

Values represent averages per node. Outliers values are the exact numbers of outliers detected. Times are expressed as minutes.

6 Conclusions

In this paper we presented SMART a system for effective monitoring of enterprise activities based on RFID data. We showed our notion of outliers and a suitable template for modeling them in an effective way by the enterprise management. A case study concerning the use of our system in a real life application scenario is presented and it shows the high quality of the results obtained exploiting SMART system for shipping monitoring. The system is subject to further improvement, in particular we plan as a first step to include additional outlier functions.

References

1. Avnur, R., Hellerstein, J.M.: Eddies: Continuously adaptive query processing. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, Dallas, Texas, USA, pp. 261–272 (2000)
2. Datar, M., Motwani, R., Babcock, B., Babu, S., Widom, J.: Models and Issues in Data Stream Systems. In: Twenty-first ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, Madison, Wisconsin, USA, pp. 1–16 (2002)
3. Li, X., Gonzalez, H., Han, J., Klabjan, D.: Warehousing and Analyzing Massive RFID Data Sets. In: Proc. of the ICDE Conference (2006) (to appear)
4. Madden, S., Hellerstein, J.M.: Distributing queries over low-power wireless sensor networks. In: ACM SIGMOD Int. Conf. on Management of Data, Madison (WI), USA (2002)
5. Raghavan, P., Henzinger, M.R., Rajagopalan, S.: Computing on data streams. Technical Report 1998-011, Digital Systems Research Center (1998), `http://www.research.digital.com/SRC/`
6. Gehrke, J., Bonnet, P., Seshadri, P.: Querying the Physical World. IEEE Personal Communication 7 (2000)

7. Alert System, http://www.alertsystems.org
8. Li Lee, M., Ganti, V., Ramakrishnan, R.: ICICLES: Self-tuning Samples for Approximate Query Answering. In: Proceedings of 26th International Conference on Very Large Data Bases, Cairo, Egypt, pp. 176–187 (2000)
9. Thakkar, H., Wang, H., Bai, Y., Luo, R.C., Zaniolo, C.: An introduction to the Expressive Stream Language (ESL). Tech. Report
10. Pei, J., Chen, Q., Liu, Y., Chen, L., Zhao, Y.: Mining frequent trajectory patterns for activity monitoring using radio frequency tag arrays. In: PerCom, pp. 37–46 (2007)

Quality–Driven Query Processing Techniques in the MOMIS Integration System

Domenico Beneventano and R. Carlos Nana Mbinkeu

DII - Università di Modena e Reggio Emilia
Via Vignolese 905, 41100 Modena- Italy
firstname.lastname@unimore.it

Abstract. In the NeP4B project the MOMIS data integration system was extended with data quality concepts. Starting from this obtained framework, the aim of this paper is twofold; first, we consider data quality metadata in the specification of queries on the Global Schema of the integration system; in this way we can express the quality of the retrieved data by means of threshold of acceptance. Second, we will discuss how the quality constraints specified in the query are useful to perform some query optimization techniques.

1 Introduction

MOMIS (Mediator envirOnment for Multiple Information Sources) is a framework to perform integration of structured and semi-structured data sources [10,11,13,12,4]; the MOMIS integration process gives rise to a Global Virtual View (GVV), in the form of global classes and global attributes, which represents a unified and integrated view of data residing in the different local data sources. MOMIS is characterized by a classical wrapper/mediator architecture: the data sources contain the real data, while the GVV provides a reconciled, integrated, and virtual view of the underlying sources. The MOMIS query management environment is able to process incoming queries on the GVV.

In the context of the Nep4B project (http://www.dbgroup.unimo.it/nep4b) the MOMIS system was extended with data quality concepts [6]; the issue of data quality is gaining an increasingly important role in the context of information integration [1]; with respect to other aspects of information integration, such as schema mapping, which have been extensively studied, this issue requires further investigation.

Starting from this obtained framework, the aim of this paper is twofold; first, we consider data quality metadata in the specification of queries on the GVV ; in this way we can express the quality of the retrieved data by means of threshold of acceptance; in particular, we will consider Data Quality Aware Queries, where data quality metadata are used in the specification of queries on the GVV.

Second, we will discuss how the quality constraints specified in the query are useful in a Quality-Driven Query Processing to perform some query optimization techniques.

The rest of this paper is organized as follows. Section 2 introduces the basic definition of the MOMIS Data Integration System. In Section 3 the MOMIS framework is extended to allow data at sources to be annotated with data quality metadata. Section 4 introduces Data Quality Aware Queries; in particular, in section 4.1 we will discuss how the quality constraints specified in the query are useful to perform some query optimization techniques.

2 The MOMIS Data Integration System

In this section we will introduce the basic definition of the MOMIS framework that will be used along this paper. A MOMIS Data Integration System is constituted by:

– A Global Virtual View GVV.
– A set $\mathcal{N}$ of local data sources with the related local schema.
– A set $\mathcal{M}$ of GAV (Global-As-View) mapping assertions between GVV and $\mathcal{N}$, where each assertion associates to a class G in GVV a mapping query over a set of local sources in $\mathcal{N}$. More precisely, for each global class $G \in GVV$ we define a set of local classes $L(G) \in \mathcal{N}$ and a mapping query MQ_G over the schemas of local sources in $\mathcal{N}$.

Intuitively, the GVV is the intensional representation of the information provided by the Integration System, whereas the mapping assertions specify how such an intensional representation relates to the local sources managed by the Integration System. Both the GVV and the local source schemata are expressed in the ODL_{I^3} language [5]. However, for the scope of this paper, we consider both the GVV and the local source schemata as relational schemata.

As described in [3], the GVV and the mapping assertions are defined in a semi-automatic way (i.e the integration designer is supported by the system) as follows:

1. The system detects semantic similarities among the involved source schemas and automatically generates a GVV (constitued by global classes and global attributes); moreover, for each Global Class G in the GVV, the system defines the local classes $L(G)$ belonging and a Mapping Table (MT), whose columns represent $L(G)$ and whose rows represent the global attributes of G. An element $MT[GA][L]$ represents the local attribute of L which is mapped onto the global attribute GA. As example we consider the following two local classes (relations) of two different local sources:

```
resort(name, rooms, amount, stars)
hotel(name, hotelrooms, price, wifi)
```

Starting from these local classes, a GVV with the global class `Hotel` and the related Mapping Table of Figure 1 are automatically generated.

Hotel	**resort**	**hotel**
`Name`	`name`	`name`
`Room`	`rooms`	`hotelrooms`
`Price`	`amount`	`price`
`Stars`	`stars`	-
`Wifi`	-	`wifi`

Fig. 1. Mapping Table of global class Hotel

2. Join Conditions are defined, by the integration designer, between pairs of local classes belonging to G in order to identify instances of the same real-world object in different sources.

   In our example, we specify as join condition `resort.name=hotel.name` intending that instance of the classes resort and hotel having the same name represent the same real object.
3. Resolution Functions [14,15] are introduced for global attributes to solve data conflicts of local attribute values associated to the same real-world object. In [14] several resolution functions are described and classified; in our framework we consider and implement some of such resolution functions, in particular, the PREFERRED function, which takes the value of a preferred source and the RANDOM function, which takes a random value. If the designer knows that there are no data conflicts for a global attribute mapped onto more than one source (that is, the instance of the same real object in different local classes have the same value for this common attribute), he can define this attribute as an Homogeneous Attribute; of course, for homogeneous attributes resolution functions are not necessary. A global attribute mapped into only one local class is a particular case of an homogeneous attribute. Attributes `Price` and `Room` are non-homogeneous therefore it is necessary to apply a resolution function. In our case, we apply an `AVG` function on attribute `Price` and `CONCATENATE` function on attribute `Room`.
4. The mapping query for a global class G is obtained by performing the full outerjoin-merging of the local classes $L(G)$ belonging to G; from an extensional point of view, the mapping query defines the instance of a global class starting from the instances of local classes. The full outerjoin-merge operator was introduced in [15] and is defined in the MOMIS framework in [7]; intuitively, it corresponds to the following two operations:
   (a) Computation of the full outerjoin on the basis of the join conditions;
   (b) Application of the Resolution Functions

   As an example, by considering the instances of `resort` and `hotel` show in figure 2, the result of the full outerjoin-merge between `resort` and `hotel`, is in figure 3. For the local attributes `resort.name` and `hotel.name` which are used in join condition, we do not consider values for data quality dimension, since this quality information is not used in our method.

resort

name	stars	amount	rooms
Hilton	5	120	12
Kyriad	5	350	NULL
Sofitel	NULL	210	40
Ibis	4	100	6
Garden	4	100	24
Continental	3	225	122

hotel

name	price	hotelrooms	wifi
Hilton	80	13	false
Kyriad	400	4	true
Garden	100	18	NULL
Sofitel	100	6	true
Madison	100	NULL	false
Dream	200	35	true

Fig. 2. Instances of local classes `hotel` and `resort`

Hotel

Name	Stars	Price	Rooms	Wifi
Hilton	5	100	12, 13	false
Kyriad	5	375	4	true
Sofitel	NULL	160	6, 40	true
Ibis	4	100	6	NULL
Garden	4	100	18, 24	NULL
Continental	3	225	122	NULL
Madison	NULL	100	NULL	false
Dream	NULL	200	35	true

Fig. 3. Instances of global class Hotel computed as the Full outerjoin-merge between `hotel` and `resort`

3 Data Quality Dimensions in MOMIS

In the Nep4B project the MOMIS system was extended with data quality concepts [6]; on the basis of these results, we assume a set QD of quality dimensions with the following hypothesis:

1. As in [13], we consider that each quality dimension $QDim$ in QD is associated with a domain of possible values, and a total order is assumed to be defined on the domain. All quality dimensions are normalized: each quality dimension value is linearly mapped to a number in the interval [0,1]. The mapping is done so that high quality dimension values are always more desirable than low values; that is, the worst quality dimension value is mapped to 0, and the best to 1.
2. As in [19], quality dimensions are defined at the source attribute level: every local attribute LA has a subset `QD_GA` of $QDim$ associated with it. Starting from the quality dimensions associated with local attributes, for each global attribute GA of a global class G is defined a set of quality dimensions `QD_GA` as the union of the quality dimensions of the local attributes corresponding to GA in the mapping table of G.
   As an example, we will use as quality dimensions the following set:

$$QD = \{\texttt{accuracy}, \texttt{completeness}, \texttt{consistency}, \texttt{currency}\}$$

Hotel	**resort**	**hotel**
`Name`	`name`	`name`
`Room`	`rooms`	`hotelrooms`
`accuracy`	`0.8`	`0.9`
`Price`	`amount`	`price`
`accuracy`	`0.7`	`0.8`
`completeness`	`0.8`	`0.9`
`consistency`	`NULL`	`0.8`
`Stars`	`stars`	`-`
`accuracy`	`0.6`	
`completeness`	`0.5`	
`consistency`	`0.3`	
`currency`	`0.7`	
`Wifi`	`-`	`wifi`
`consistency`		`0.5`

Fig. 4. Mapping Table and Quality Mapping Table of the global class `Hotel`

where `accuracy` stands for accuracy, `completeness` for completeness, `consistency` for consistency and `currency` for currency; moreover, we consider:

`QD_resort.amount = {accuracy, completeness}` and
`QD_hotel.price = {accuracy, completeness, consistency}`

Then for the global attribute `Price` we obtain:

`QD_Price = {accuracy, completeness, consistency}`

The quality dimension values for local attributes are represented by means of a *Quality Mapping Table* MTQ associated to a global class G: MTQ is a table whose columns represent the local classes $L(G)$ belonging to G and whose rows represent the quality dimensions defined for each global attributes A of G. An element $MTQ[A.QDim][L]$ represents the value of $QDim$ for the local attribute $LA = MT[A][L]$ of L, or is `NULL` if $QDim$ is not associated to LA, i.e. if $QDim \notin QD\_LA$; as an instance, in our example $MTQ[\texttt{Price.cons}][\texttt{resort}] = \texttt{NULL}$. To give an immediate representation of quality dimensions we represent both the Mapping Table MT and the Quality Mapping Table MTQ in the same table, as in figure 4 (when $MT[A][L]$ is NULL, $MTQ[A.QDim][L]$ doesnt exists: in the figure, this is denoted by a empty cell).

4 Data Quality Aware Queries in the MOMIS

In this section we introduce Data Quality Aware Queries in the MOMIS framework: in the specification of a *global query* on the GVV, quality dimensions can be used to express the quality of the retrieved data.

The term quality-aware queries is introduced in [19] where the authors propose to include user specific quality considerations into query formulations, in order to address user specific requirements. The approach of [19] is based on generic Collaborative Information Systems. A more specific approach in the context of Data Integration Systems is proposed in [13], where queries on the integrated and global schema are able to express the quality of the retrieved data by means of threshold of acceptance with which users can ensure minimal performance of the data. Other mechanisms by which queries can be expressed over data and quality metadata have been proposed [17,16].

In this paper we adopt the proposal of [13]: quality dimensions are used to specify `quality constraints`[1] that express the quality of the retrieved data, by means of threshold of acceptance, excluding from the answer data that do not satisfy the specified quality constraint.

A global query with *quality constraints* is a conjunctive query on a global class G of the GVV expressed in an extended SQL syntax as follow:

```
select [restrictive] $A_1, \ldots, A_k$
from $G$
    where $pred_1$ and ... and $pred_n$
    using $Qpred_1$, ..., $Qpred_m$
```

where A_i are global attributes of G, *pred* is a (simple) predicate : A *op value* *Qpred* is a *quality constraint*: $A.QDim > Qvalue$, where $QDim$ is a quality dimension defined for the global attribute A and $Qvalue$ is a value in [0, 1].

Example 1. Let us consider the query Q1 without quality constraints.

```
Q1 = select Name, Room
     from Hotel
     where Price = 100 and Stars > 3
```

By considering the instance of `Hotel` of figure 3, the answer of this query is:

Name	Room
Hilton	12, 13
Ibis	6
Garden	24,18

The `using` clause implements a selection predicate for specifying the desired quality criteria of the answer. In analogy with the `where` clause, which restricts the answer set with a condition on attributes, the `using` clause restricts the answer set with a condition on quality dimensions. Let us give an intuitive explanation of the semantics of the `using` clause; a deep discussion about quality constraints is in [13]. Let A be a global attribute, L a local class and $LA = MT[A][L]$ the local attribute of L mapped into A. Since a quality dimension $QDim$ is defined at the local attribute level, the quality constraint ($A.QDim$ op $Qvalue$), specifies that the values of LA will be considered in the answer of the query only if the quality dimension $QDim$ associated with LA is greater than the desired $Qvalue$, i.e. if $MTQ[A.QDim][L] > Qvalue$. In this case we will say that the quality constraint is satisfied by LA. As discussed in

[1] In [13] the term *quality predicate* is used.

[13] , when $MTQ[A.QDim][L]$ is `NULL` the quality constraint is satisfied by LA, i.e the values of LA will be considered in the answer of the query, only if **restrictive** is specified.

Example 2. Let Q2 obtained from Q1 by adding some quality constraints:

```
Q2 = select Name, Room
     from Hotel
     where Price = 100 and Stars > 3
     using Price.acc > 0.7, Price.cons > 0.6, Room.acc > 0.7
```

The result of the query Q2 is:

Name	Room
Garden	24,18

where we can observe that there is no longer (w.r.t the Q1 answer) a Hilton record. Since the quality constraint `Price.acc > 0.7` is not satisfied by the local attribute `amount`, the values of this local attribute are not considered to calculate the value of the `Price` global attribute: then the value of `Price` for Hilton record is equal to 80 and thus the constraint `Price = 100` is false for this record. For the same reason the Ibis record is not in the Q2 result.

4.1 Quality-Driven Query Processing

To answer a global query on G, the query must be rewritten as an equivalent set of queries (*local queries*) expressed on the local classes $L(G)$ belonging to G. This query rewriting is performed by considering the mapping between the GVV and the local schemata; in a GAV approach the query rewriting is performed by means of *query unfolding*, i.e., by expanding the global query on G according to the definition of its mapping query MQ_G.

The query unfolding of a global query Q produces, for each local class L belonging to G, a local query `Q_L`

```
Q_L = select S_L
      from L
      Where C_L
```

where the select list `S_L` is a list of local attributes of L and the condition `C_L` is a conjunction of (simple) predicates on L. These local queries are executed on the local sources and local queries answers are then fused by means of the mapping query MQ_G to obtain the answer of the global query.

We defined this query execution process in several papers [2,4]. The novelty contribution of this paper is to describe how quality constraints are used in the query unfolding process. In particular, we will show how quality constraints are useful to perform query optimization.

The process for executing the global query consists of the following steps:

1. **Computation of Local Query predicates:**
   In this step is evaluated whether a predicate of the global query may be pushed down to the local level, i.e. if it may be into the local query condition for a local class; this decision also depends on the presence of quality constraints. A predicate (GA *op value*) is translated into the predicate ($MT[GA][L]$ *op value*) on the local class L only if:

   (1) $MT[GA][L]$ is not null **and**
   (2) all quality constraints on GA are satisfied by $MT[GA][L]$ **and**
   (3.a) GA is an homogeneous attribute **or**
   (3.a) for any other local attribute LA different from $MT[GA][L]$ where GA is mapped, there are quality constraints on GA not satisfied from LA.

   Condition (1) and (2) are straightforward; conditions (3.a) and (3.b) are discussed in the following by considering queries Q1 and Q2, respectively.

   For the query Q1, since the global attribute `Stars` is mapped only in resort, and then it is homogeneous, the predicate `(Stars > 3)` is pushed down for the local class `resort`. On the other hand, the global attribute `Price` is not homogeneous (it is defined by means of the `AVG` function) and then the predicate `(Price = 100)` cannot be pushed at the local level: since the `AVG` function is calculated at a global level, the predicate may be globally true but locally false. Thus, for the query Q1 we obtain the following local query conditions:

   `C_hotel: true` (i.e. the local query does not have a where condition)
   `C_resort: stars > 3`

   For query Q2 the translation of the predicate `(Stars > 3)` doesn't change w.r.t. query Q1 since for `Stars` there are no quality constraints specified. In the translation of the predicate `(Price = 100)` on the local class `hotel` condition (3.b) is true: since the quality constraint `(Price.acc > 0.7)` is not satisfied by the other local attribute `resort.amount`, only values coming from `hotel` must be considered in the evaluation of predicate `(Price = 100)`, as discussed above; then this predicate can be pushed down on `hotel`. In this way, for query Q2 we obtain the following local query conditions:

   ```
   C_hotel: price = 100
   C_resort: stars > 3
   ```

   In this way the presence of a quality constraint allows to perform the push down of the predicate `(Price = 100)` on local class `hotel`.
2. **Computation of Residual Conditions:** A predicate on a global attribute GA that in step (1) is pushed down on all local classes such that $MT[GA][L]$ is not null, is already solved, i.e. in the full join of the local queries this predicate is already satisfied; otherwise the predicate is considered as residual and have to be solved at the global level.

For the query Q1, the computation of residual conditions gives: `Price = 100` while for query Q2 there are no residual predicates.

3. **Generation and execution of local queries:** The select list S_L is obtained as $X - Y$ where
   (a) X = union of the attributes of the global select list, of the Join Condition and of the Residual Condition; these attributes are transformed into the corresponding ones at the local level on the basis of the Mapping Table.
   (b) Y = union of local attribute $LA = MT[A][L]$ such that there is a quality constraint on A not satisfied from LA.

   For instance, for the query Q2 since `(Price.acc > 0.7)` is not satisfied by `resort.amount`, this local attribute can be eliminated from the select list of Q_resort. In other words, the presence of quality constraints may allow the elimination of attributes from the select list of a local query: this corresponds to a query optimization technique (called *projection reduction*) since it reduces the size of the local query answer.

   With the step 3, query unfolding ends and local queries are generated; for the query Q1 we obtain

```
LQ_hotel_1  = select name, price, hotelrooms  from hotel

LQ_resort_1 = select name, amount, rooms from resort
              where stars > 3
```

   and for the query Q2 we obtain

```
LQ_hotel_2  = select name, hotelrooms from hotel
              where price = 100

LQ_resort_2 = select name, rooms from resort
              where stars > 3
```

4. **Fusion of local answers:** The local answers are fused into the global answer on the basis of the mapping query MQ_G defined for G. As intuitively discussed in Section 2, MQ_G is defined by using a Full Outerjoin-merge operation which, given the local answer LQ_L1 and LQ_L2, is substantially performed in two steps:
   (a) Computation of the **full outerjoin** FOJ on the basis of the join condition JC(L1,L2) defined between L1 e L2

```
FOJ = LQ_L1 full outerjoin LQ_L2 on JC(L1,L2)
```

   (b) **Application of the Resolution Functions:** for each pairs of attributes in FOJ (except for attributes used in the join condition) mapped in the same global attributes the related Resolution Function is applied. For example, in query Q1, we need to apply the resolution function `AVG` and `CONCATENATE` respectively to global attribute `Price` and `Room`.

In step 4.a the computation of the full outerjoin operation can be optimized by reducing it to a left/right or inner join on the basis of the following rules:

(1) `FOJ = LQ_L1 left join LQ_L2 on JC(L1,L2)`
if there exists a predicate pushed down only on L2

(2) `FOJ = LQ_L1 inner join LQ_L2 on JC(L1,L2)`
if there exists a predicate pushed down only on L2 and a predicate pushed down only on L1

For the query Q1, since the predicate `Stars > 3` is pushed down only in `resort`, rule (1) holds and then *FOJ* can be computed as

```
FOJ_1 = LQ_resort_1 as R1 left join LQ_hotel_1 as H1
        on (R1.name = H1.name)
```

For the Query Q2, besides the predicate `Stars > 3`, is pushed down only in `resort`, we have `Price = 100` pushed down only in `hotel`, then rule (2) holds and thus *FOJ* can be computed by using inner join:

```
FOJ_2 = LQ_resort_2 as R2 inner join LQ_hotel_2 as H2
        on (R2.name = H2.name)
```

The substitution of a full outer join with a left/right outer join or a join constitutes a relevant query optimization result, since full outer join queries are very expensive, especially in a distributed environment as the one of mediator/integration systems. Moreover, while database optimizers take full advantage of associativity and communtativity properties of join to implement efficient and powerful optimizations on join queries, only limited optimization is performed on full outer join [9]. The presence of quality constraints in the global query may contribute to perform this kind of optimization, as shown for the query Q2.

5. **Application of the Residual Condition:** the result of the global query is obtained by applying the residual condition to the `R_FOJ`.
In our example, for the query Q1 we have `Price = 100` as residual condition then its result is obtained as

```
select Name, Room from R_FOJ_1  where Price = 100
```

while for the query Q2, we have no residual condition then its result is obtained as `select Name, Room from R_FOJ_2`.

5 Conclusions and Future Work

In this paper, we presented our current work to extend the MOMIS Data Integration System with Data Quality Information; in particular, we introduced data quality metadata in the specification of queries on the Global Schema, by considering Data Quality Aware Queries. An interesting result is that quality constraints specified in the query are useful to perform some relevant query optimization techniques, such as predicate push down, projection reduction and fulf outer join simplification.

Our future work will consist in investigate how quality metadata can be used in other important aspects of a Data Integration System. First of all, we will evaluate how quality metadata can be exploited to define quality-based resolution function and how these new resolution functions affect the query processing. To this end, we will consider the approach to conflict resolution adopted in the Fusionplex (which is similar to the one used in the DaQuinCIS [17], as discussed in in [6]): to resolve attribute conflicts on the basis of quality metadata associated with data of local sources.

Moreover, we will further investigate Data Quality Aware Queries, by considering new kind of quality constraints and how the Quality-driven Query Processing need to be extended for these new quality constraints.

Finally, we will evaluate the definition of quality dimension associated to a global schema level, since these quality metadata can be useful for a high level integration process, when an integration system need to expose its information knowledge to other integration systems (i.e. in a peer-to-peer system where global classes can exchange information mutually). The computation of quality metadata for the global schema needs a strategy to aggregate the value metadata of multiple local sources to obtain a single value that summarizes all the values.

References

1. Batini, C., Scannapieco, M.: Data Quality: Concepts, Methodologies and Techniques, Data-Centric Systems and Applications. Springer, New York (2006)
2. Beneventano, D., Bergamaschi, S.: Semantic search engines based on data integration systems. In: Semantic Web Services: Theory, Tools and Applications. Idea Group, USA (2006) (to appear), http://dbgroup.unimo.it/SSE/SSE.pdf
3. Beneventano, D., Bergamaschi, S., Guerra, F., Vincini, M.: Synthesizing an integrated ontology. IEEE Internet Computing 7(5), 42–51 (2003)
4. Beneventano, D., Bergamaschi, S., Vincini, M., Orsini, M., Mbinkeu, R.C.N.: Query translation on heterogeneous sources in momis data transformation systems. In: VLDB 2007 - International Workshop on Database Interoperability (InterDB 2007), Vienna, Austria, September 24 (2007)
5. Bergamaschi, S., Castano, S., Vincini, M., Beneventano, D.: Semantic integration of heterogeneous information sources. Data Knowl. Eng. 36(3), 215–249 (2001)
6. Consortium, T.N.: Revised specification for building semantic peer. Nep4b - networked peers for business, deliverable d2.2, Dipartimento di Ingegneria dell'Informazione (2009), http://dbgroup.unimo.it/TechnicalReport/NeP4BD2.2.pdf
7. Beneventano, D., et al.: Detailed design for building semantic peer, nep4b technical report. Technical report, Dipartimento di INgegneria dell'Informazione (2008), http://www.dbgroup.unimo.it/nep4b/risorse/d2.1.m20.pdf
8. Huh, Y., et al.: Data quality, information and software technology. Information and Software Technology 32(8), 559–565 (1990)
9. Galindo-Legaria, C.A., Rosenthal, A.: Outerjoin simplification and reordering for query optimization. ACM Trans. Database Syst. 22(1), 43–73 (1997)
10. Garcia-Molina, H., et al.: The TSIMMIS approach to mediation: Data models and languages. In: NGITS workshop (1995), ftp://db.stanford.edu/pub/garcia/1995/tisimmis-models-languages.ps

11. Genesereth, M.R., Keller, A.M., Duschka, O.: Infomaster: An Information Integration System. In: Proceedings of 1997 ACM SIGMOD Conference (1997)
12. Halevy, A.Y.: Answering queries using views: A survey. VLDB J. 10(4), 270–294 (2001)
13. Motro, A., Anokhin, P.: Fusionplex: resolution of data inconsistencies in the integration of heterogeneous information sources. Inf. Fusion 7(2), 176–196 (2006)
14. Naumann, F., Bleiholder, J.: Conflict handling strategies in an integrated information system. In: Proceedings of the WWW Workshop in Information Integration on the Web, IIWEB (2006)
15. Naumann, F., Freytag, J.C., Leser, U.: Completeness of integrated information sources. Inf. Syst. 29(7), 583–615 (2004)
16. Naumann, F., Leser, U., Freytag, J.C.: Quality-driven integration of heterogenous information systems. In: VLDB 1999, pp. 447–458 (1999)
17. Scannapieco, M., Virgillito, A., Marchetti, M., Mecella, M., Baldoni, R.: The daquincis architecture: a platform for exchanging and improving data quality in cooperative information systems. Inf. Syst. 29(7), 551–582 (2004)
18. Wang, R.Y., Reddy, M.P., Kon, H.B.: Toward quality data: an attribute-based approach. In: Special Issue on Information Technologies and Systems, March 1995, pp. 349–372. Elsevier Science Publishers, Amsterdam (1995)
19. Yeganeh, N.K., Sadiq, S.W., Deng, K., Zhou, X.: Data quality aware queries in collaborative information systems. In: APWeb/WAIM, pp. 39–50 (2009)

Towards a Model for the Multidimensional Analysis of Field Data

Sandro Bimonte[1] and Myoung-Ah Kang[2]

[1] CEMAGREF, Campus des CEZEAUX,
63173 AUBIERE, France
[2] LIMOS-UMR CNRS 6158,ISIMA, Blaise Pascal University, Campus des CEZEAUX,
63173 AUBIERE, France
`Sandro.bimonte@cemagref.fr, kang@isima.fr`

Abstract. Integration of spatial data into multidimensional models leads to the concept of Spatial OLAP (SOLAP). Usually, SOLAP models exploit discrete spatial data. Few works integrate continuous field data into dimensions and measures. In this paper, we provide a multidimensional model that supports measures and dimension as continuous field data, independently of their implementation.

Keywords: Spatial OLAP, Field data, Spatial Data Warehouses, Multidimensional models.

1 Introduction

It has been estimated that about 80% of the data stored in corporate databases integrates geographic information [7]. This information is typically represented according two models, depending on the nature of data: *discrete (vector)* and *field* [22]. The latter model represents the space as a continuous field. Fields have to be discretized to be represented into computers according to data input, and data analysis. These representations can be grouped into two categories: *incomplete* and *complete*. Incomplete representations store only some points and need supplementary functions to calculate the field in non-sampled areas. Complete representations associate estimated values to regions and assume that this value is valid for each point in the regions (raster). Fields are very adapted for modeling spatial phenomena such as pollution, temperature, etc. They allow a point by point analysis through the Map Algebra operators [16] [22].

In order to benefit from Data warehousing and OLAP decision support technologies [11] also in the context of spatial data, some works extended them leading to the concept of Spatial OLAP (SOLAP), which integrates OLAP and Geographic Information Systems (GIS) functionalities into a unique framework [2]. As for the model underlying SOLAP systems, several research issues from theoretical and implementation point of view have risen. Indeed, several works focus on indexing [20] and visualization techniques [6]. Motivated by the relevance of a formal representation of SOLAP data and operators, some spatio-multidimensional models based on vector data have been defined (a review can be found in [4]). Few works consider field data into multidimensional models [1], [23] and [15]. These models present some limitations that do not allow fully exploiting continuous field data into OLAP from these points of view: *aggregation functions*, *hierarchies based on field*, and *independence of field data implementation*. Thus, in this paper we propose a spatio-multidimensional model integrating field data.

The reminder of this paper is organized as follows: Section 2 recalls fundamentals of spatial analysis techniques, and existing SOLAP models for field data. Our model is proposed in the Section 3. Finally, in Section 4 conclusions and future work are presented.

2 Related Work

The term Map Algebra was first introduced in [22] to describe operators on raster data (complete field data). Map Algebra operators are classified according to the number of grids and cells involved. *Local operators* apply a mathematical or logical function to the input grids, cell by cell. *Focal operators* calculate the new value of each cell using neighboring cells values of the input grid. *Zonal operators* calculate new values as a function of the values of the input grid which are associated with the zone of another grid, called zone layer. An extension of the Map Algebra (Cubic Map Algebra) to the temporal dimension is presented in [16]. The authors redefine Map Algebra operators on a cube of cells whose coordinates are three-dimensional. Then, the sets of operators are modified accordingly as shown on Figure 1. In [5] the authors define an algebraic model for field data. This framework provides a formal definition of Map Algebra operators independent of their implementation. Along this line, the work of [8] formally describes how Map Algebra fundamentals can be used for image manipulation. Finally, Map Algebra has been defined also for incomplete field data such as Voronoi tessellations [13]. Indeed, "*Map Algebra obliges analysts to organize reality according to a particular data structure (raster) instead of allowing making the reality suggest the most adequate data structure for the analysis*" [10].

On the other hand, the integration of spatial data into data warehousing and OLAP systems leads to the concept of Spatial OLAP (SOLAP). SOLAP is based on the spatial multidimensional model that defines *spatial dimensions* and *spatial measures*. In particular, a spatial dimension is a classical dimension whose levels contain spatial attributes. This allows for visualizing measures on maps to discover (spatial) relations and unknown patterns. According to the vector model, a spatial measure is defined as a set of geometries and/or the result of spatial operators [2] [14]. Some spatial multidimensional models, based on the vector model, have been proposed [4]. Despite of the significant analysis power of field data and its associated spatial analysis operators, only few works address this issue. [15] extends the concepts of spatial dimension to informally define "spatial matrix (raster) dimension" where at least one level represents raster data. Then, a member is a cell of the raster. Moreover, she introduces also the concept of "matrix cube" where each fact is associated to a cell of the raster data with its attributes. Aggregation functions are Map Algebra operators (local, focal and zonal). The author limits field data to raster. By this way, the work loses in terms of abstraction for field data. Indeed, Map Algebra has been defined also on other forms of representation of field data such as Voronoi tessellation [13], and terrain representations, such as TIN, that could be very useful for understanding and analyzing multidimensional data [6].

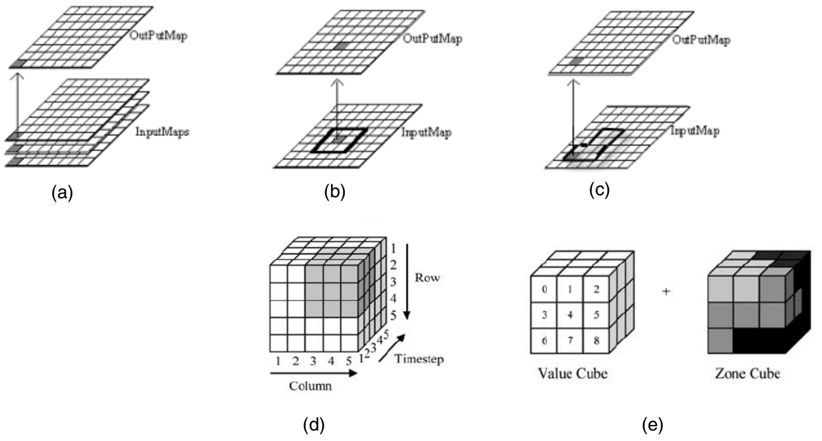


Fig. 1. Map Algebra: a) local, b) focal, c) zonal [22], Cubic Map Algebra: d) focal, e) zonal [16]

Hence, [23] proposes a conceptual multidimensional model for taking into account field data independently of their implementation. They define a "field measure" as a value that changes in space and/or time. The model limits aggregation functions for field measure to local Map Algebra functions. They also introduce a "field hierarchy" when a level is a field data that is not linked to facts. Consequently, it is not possible to have field measures at different granularities, which are mandatory for spatial analysis [21]. Finally [1] defines the concept of "continuous cube" when spatial dimensions are composed of infinite spatial members whose associated numerical measures are calculated using interpolation functions, as they use an incomplete representation of spatial members. This model does not allow introduce field data as measures and in hierarchies.

Therefore, a formal model is necessary to create the foundation for a framework for the multidimensional analysis of field data as dimensions (field hierarchies) and measures. Table 1 shows the requirements for this model and how existing works address them. As shown in table 1, there is no existing model that supports all of these requirements. This is the reason why we propose a new SOLAP model in this paper.

Table 1. Requirements for spatial multidimensional model for continuous field data

Requirements	[1]	[23]	[15]
Measures as continuous field data	NO	YES	Partially (only for raster data)
Hierarchy on continuous field data	NO	NO	Partially (only for raster data)
Aggregation functions as Map Algebra functions	NO	Partially (only local functions)	Partially (only for raster data)
Independence of implementation	YES	YES	NO

3 Spatio-multidimensional Model for Field Data

Before introducing our model we present a SOLAP application for monitoring earthquakes in Italian regions. Spatial analyst wants answer to queries like this: "*Where were earthquakes, and what was their intensity per region at different scales (resolutions)?*". This SOLAP application presents a measure that is a field object representing the earthquakes, a temporal dimension, and a spatial dimension that represents terrain models of Italian regions at different scales.

3.1 Geographic Data Model

In this section, we provide a uniform representation for field and vector data, which are used by the multidimensional model to define measures and dimensions members.

An Object represents an object of the real world described by some alphanumeric attributes. It is used in the model to represent levels and members (Sec. 3.2).

Definition 1. Object

An Object Structure S_e is a tuple $\langle a_1, \ldots a_n\rangle$ where $\forall i \in [1,\ldots n]$ a_i is an attribute defined on a domain $dom(a_i)$

An Instance of an Object Structure S_e is a tuple $\langle val(a_1),\ldots val(a_n)\rangle$ where $\forall i \in [1,\ldots n]$ $val(a_i) \in dom(a_i)$

We denote by '$I(S_e)$' the set of instances of S_e

A Geographic Object extends an Object to represent geographic information according to the vector model. Indeed, a Geographic Object [3] is a geometry (*geom*) and an optional set of alphanumeric attributes (*$[a_1, \ldots a_n]$*) whose values are associated to the whole geometry according to the vector model (Figure 2a).

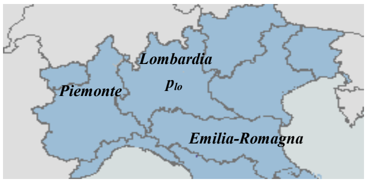
Definition 2. Geographic Object

Let $g \subset R^2$ i.e. a subset of the Euclidian space. An Object Structure $S_e = \langle geom, [a_1, \ldots a_n]\rangle$ is a Geographic Object Structure if the domain of the attribute $geom$ is a set of geometries: $dom(geom) \in 2^g$

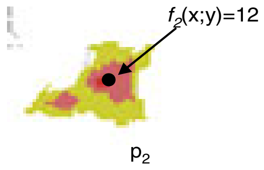
$geom$ is called 'geometric support'

Example 1

The geographic object structure representing Italian regions is $S_{region}=\langle$geom, name$\rangle$ where 'geom' is the geometric support, and 'name' is the name of the region. An instance of S_{region} is $t_2 = \langle p_{lo}$, Lombardia$\rangle$ where 'p_{lo}' is the geometry of the region Lombardia (Figure 2a).



(a)



(b)

Fig. 2. a) Italian regions: Instances of S_{region}, b) Earthquakes: an instance of S_{earthq}

According to [13] fields are geographic objects with a function that maps each point of their geometry to an alphanumeric value. This definition allows for representing field data independently of their implementation (complete/incomplete) (Field Object). Thus, A Field Object extends a Geographic Object with a function that associates each point of the geometry to an alphanumeric value. In this way, a Field Object allows for representing geographic data according to the field and the vector model at the same time ("Independence of implementation" requirement of Table 1).

Definition 3. Field Object

Let $S_e = \langle geom, field, [a_1, \ldots a_n] \rangle$ a Geographic Object Structure. S_e is a Field Object Structure if the domain of the attribute field is a set of functions defined on m sub-sets of points of geom having values in an alphanumeric domain dom_{field} : $dom(field) = \{f_1 \ldots f_m\}$

An Instance of an Field Object Structure S_e is a tuple $\langle g, f_j, val(a_1), \ldots val(a_n) \rangle$ where:

- $\forall i \in [1, \ldots n]\ val(a_i) \in dom(a_i),\ g \in dom(geom)$
- $f_j : g \rightarrow dom_{field}$ and $f_j \in \{f_1, \ldots, f_m\}$

We note 'field support' the input domain of f_j

Example 2

The field object structure representing earthquakes is $S_{earthq}=\langle geom, intensity\rangle$ where 'geom' is the geometric support, and 'intensity' is a set of functions defined on 'geom' with values in *R*. 'intensity' represents the intensity of the earthquake. An instance of S_{earthq} is $t_2 = \langle p_2, f_2 \rangle$ where p_2 is a geometry and f_2 represents the intensity of the earthquake on the 11-1999 in Lombardia. f_2 is defined on each point of p_2 with values in *R*, for example $f_2(x;y) = 12$ (Figure 2b).

By the same way, we can define a field object structure to represent terrain models of Italian regions at different scales by adding to the geographic object S_{region} the field attribute representing terrain elevation.

3.2 Spatio-multidimensional Model for Field Data

A spatio-multidimensional model organizes data using the concepts of dimensions composed of hierarchies, and facts described by measures. An instance of the spatio-multidimensional model is a hypercube. Section 3.2.1 presents the concepts of dimensions, facts, and measures, and Section 3.2.2 formalizes cuboids.

3.2.1 Hierarchies and Facts

According to [3] a spatial hierarchy organizes vector objects in a hierarchical way. Formally, a Spatial Hierarchy organizes the Geographic Objects [3] (i.e. vector objects) into a hierarchy structure using a partial order $\leq_h$ where $S_i \leq_h S_j$ means that S_i is a less detailed level than S_j. An instance of a hierarchy is a tree ($<_h$) of instances of Geographic Objects (spatial members). Then, measures are aggregated according to the groups of spatial members defined by the tree $<_h$.

Hence, in this work we define a *Field Hierarchy* as a hierarchy of field objects. For that, we extend the spatial hierarchy by defining a tree ($<_f$) on the geometric coordinates (field supports) of the spatial members represented by field objects ("Hierarchy on continuous field data" requirement of Table 1). By this way it is possible to visualize field objects at different scales or resolutions (Figure 4). Moreover, the alphanumeric values associated to each point of the field measures are aggregated according to the groups of coordinates of spatial members defined by the tree $<_f$. By this way, our model uses the continuous representation of spatial members (Field Objects) to aggregate measure, allowing visualizing field measures at different scales or resolutions (see Figure 6b).

Definition 4. Field Hierarchy

A Field Hierarchy Structure, $\mathcal{H}_{fi}$, is a tuple $\langle \mathcal{L}_h, \lfloor_h, \lceil_h, \leq_h \rangle$ where:

- *$\lfloor_h, \lceil_h$, are of Field Object Structures, and $\mathcal{L}_h$ is a set of Field Object Structures*
- *$\leq_h$ is a partial order defined on $\mathcal{L}_h, \lfloor_h, \lceil_h$ as defined in [3]*

An Instance of a Field Hierarchy Structure $\mathcal{H}_{fi}$ is two partial orders: $<_h$ and $<_f$ such that:

- *$<_h$ is defined on the instances of $\mathcal{L}_h, \lfloor_h, \lceil_h$ as defined in [3]*
  We note $<_h$ 'geographic objects order'

- *$<_f$ is defined on the field supports of the instances of $\mathcal{L}_h, \lfloor_h, \lceil_h$ such that:*
  - *if $cood_i <_f cood_j$ then $S_i \leq_h S_j$, where $cood_i$ belongs to a field support of an instance of S_i, and $cood_j$ belongs to a field support of an instance of S_j,* ($cood_i$ and $cood_j$ are geometric coordinates)
  - *$\forall cood_i$ which does not belong to the field supports of the instances of $\lceil_h$, $\exists$ one $cood_j$ belonging to the field support of an instance of S_j such that $cood_i <_f cood_j$*
  - $\forall$ $cood_i$ which does not belong to the field supports of the instances of $\lfloor_h$, $\exists$ $cood_j$ belonging to the field support of an instance of S_j such that $cood_j <_f cood_i$

  We note $<_f$ field objects order'

The set of leafs of the tree represented by $<_h$ with root t_i are denoted as $leafs(\mathcal{H}_{fi}, t_i)$. The set of leafs of the tree represented by $<_f$ with root $cood_i$ are denoted as $leafsFieldSupport(\mathcal{H}_{fi}, cood_i)$.

Example 3

The field hierarchy structure representing the administrative dimension that groups regions into zones is $H_{location} = \langle \mathcal{L}_{location}, S_{region}, S_{all\_location}, \leq_{location} \rangle$ where $\mathcal{L}_{location} = \{S_{zone}\}$ and ($S_{region} \leq_{location} S_{zone}$). S_{region} and S_{zone} are the spatial levels of the hierarchy (Figure 3a). An example of instance of $H_{location}$ is shown on Figure 3b and 3c. We can notice two trees: the geographic objects order that is represented by black lines (Figure 3b), and the field objects order, which is represented by dashed lines (Figure 3c).

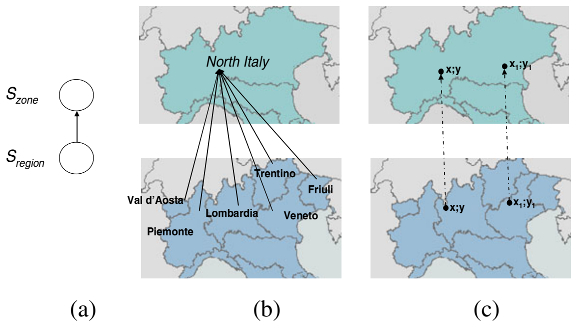


Fig. 3. Field hierarchy grouping regions into zones, a) Schema, b) Hierarchical relationships between geographic objects, c) Hierarchical relationships between geometric coordinates.

Example 4
The hierarchy structure representing the terrain models of Italian regions at different resolutions is $H_{regres} = \langle \mathcal{L}_{regres}, S_{region}, S_{all\_regres}, \leq_{regres} \rangle$ where $\mathcal{L}_{regres} = \{S_{regres}\}$ and $(S_{region} \leq_{regres} S_{regres})$ (Figure 4a). Its instance is shown on Figure 4b and Figure 4c. Note that a geometric coordinate at the coarser resolution is associated with a set of geometric coordinates at the most detailed resolution (Figure 4c). For example, this hierarchy can be defined using the bilinear interpolation algorithms used for changing resolution for raster data at different scales.

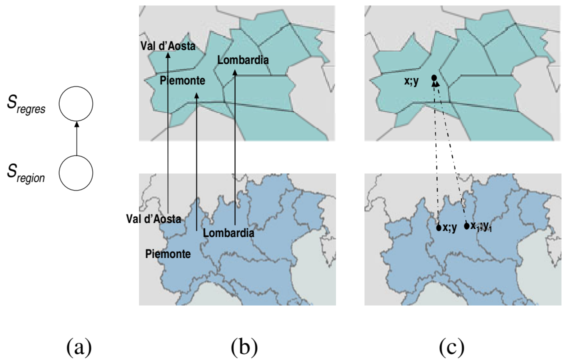


Fig. 4. Field hierarchy representing regions at different scales, a) Schema, b) Hierarchical relationships between geographic objects (geographic objects order), c) Hierarchical relationships between geometric coordinates (field objects order)

Once we have introduced the concepts of measures, and hierarchies for field data, we present the concept of Field Cube. A *Field Cube Structure* represents the spatio-multidimensional model schema where a field object is used as measure that is analyzed according classical hierarchies and one field hierarchy ("Measures as continuous field data" requirement of Table 1). Note that without losing in generalization, we suppose to have only one spatial dimension and one field measure to simplify the formalism of the model. In the same way, we do not introduce numerical measures as they can be simply represented using numeric attributes as defined in [3]. An *instance of a field cube structure* represents the fact table or the basic cuboid of the lattice cuboids.

Definition 5. Field Cube

A Field Cube Structure, $\mathcal{FC}_c$, is a tuple $\langle \mathcal{H}_1,\ldots\mathcal{H}_n, \mathcal{F}ieldObject\rangle$ where:

- *$\mathcal{H}_1$ is a Field Hierarchy Structure* (Spatial dimension)
- *$\forall i \in [2,\ldots n]$ $\mathcal{H}_i$ is a Hierarchy Structure* (Dimensions)
- *$\mathcal{F}ieldObject$ is Field Object Structure* (Field measure)

An Instance of a Field Cube Structure $\mathcal{FC}_c$, $I(\mathcal{FC}_c)$, is a set of tuples $\{\langle tb_1,\ldots tb_n, tb_f\rangle\}$ where:

- *$\forall i \in [1,\ldots n]$ tb_i is an instance of the bottom level of $\mathcal{H}_i$ ($\mathcal{L}_i$)* (Most detailed levels members)
- *tb_f is an instance of $\mathcal{F}ieldObject$* (Field measure value)

Example 5

The field cube structure of our case study is $FC_{earthq} = \langle H_{regtres}, H_{time}, S_{earthq}\rangle$. $H_{regtres}$ is the field hierarchy (the spatial dimension), H_{time} is the temporal dimension, and S_{earthq} is the field measure. It allows answering previous formulated query "*Where were earthquakes, and what was their intensity per region at different scale (resolutions)?*". Table 2 shows the instance of FC_{earthq}. Its cartographic representation is shown on Figure 5.

Table 2. Instance of FC_{earthq}

Reg	Month	Earthq
Lombardia	9-1998	t_1
Lombardia	11-1999	t_2
Piemonte	11-1999	t_6

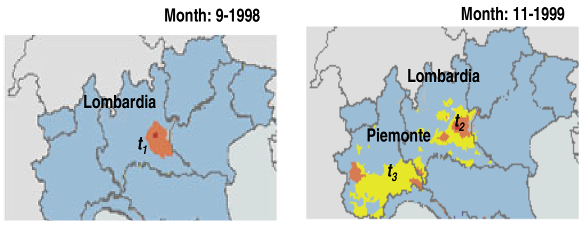


Fig. 5. Cartographic representation of the instance of FC_{earthq}

3.2.2 Hypercube

The instance of the spatio-multidimensional model is a hypercube. A hypercube can be represented as a *hierarchical lattice of cuboids* [9]. The most detailed cuboid contains detailed measures (*basic cuboid*). Other cuboids contain aggregated measures. Then, cuboids are represented by levels and (aggregated) measures values (Sec. 3.2.2.2). How field measures are aggregated from fact table data (basic cuboid) to represent non-basic cuboids is presented in Sec. 3.2.2.1.

3.2.2.1 Aggregation of Field Measures

The aggregation of field measures is defined by means of:

- For the geometric support: spatial aggregation,
- For alphanumeric attributes: alphanumeric aggregations
- For the field attribute:
  - Local Map Algebra, or focal/zonal map cubic algebra operator *when aggregating on the non-field hierarchies*
  - Alphanumeric aggregation *when aggregating on the field hierarchy*

Indeed aggregation of field measures is done in two steps. In the first step we aggregate along the non-field hierarchies, and then along the field hierarchy.

3.2.2.1.1 Aggregation on Non-Field Hierarchies

In this section we formalize the geometric and alphanumeric aggregations.

Definition 6. Spatial aggregation
Let $\mathcal{G}$ the geometric attribute. Its aggregation is defined by means of a function $O_{\mathcal{G}}$ that has as input n geometries of the attribute $\mathcal{G}$, and that returns one geometry:

$$O_{\mathcal{G}} : dom(\mathcal{G}) \times \ldots \times dom(\mathcal{G}) \rightarrow 2^{g} \text{ where } g \text{ is a subset of the Euclidian Space } R^2$$

Definition 7. Alphanumeric aggregation
Let $\mathcal{A}$ be an alphanumeric attribute. Its aggregation is defined by means of a function $O_{\mathcal{A}}$ that has in input n values of the attribute $\mathcal{A}$, and that returns one value of the attribute $\mathcal{A}$:

$$O_{\mathcal{A}} : dom(\mathcal{A}) \times \ldots \times dom(\mathcal{A}) \rightarrow dom(\mathcal{A})$$

On non-field hierarchies, the aggregation of the field attribute is defined by means of a function ($O_{\mathcal{F}}$) that takes as input a set of functions representing the field attributes values ($f_1 \ldots f_n$), and it returns a new function (f_{1n}). This function is defined on the field support of $f_1 \ldots f_n$, and the value of each point ($f_{1n}(x;y)$) is calculated by applying a alphanumeric function $O_{\mathcal{A}}$ to the values of the other functions ($O_{\mathcal{F}}(f_1(x;y) \ldots f_n(x;y))$). Then, $O_{\mathcal{F}}$ represents a map/map cubic algebra operator that is specialized in local, focal or zonal by means of the $O_{\mathcal{A}}$ function. Indeed, $O_{\mathcal{A}}$ is applied point by point for local map function, or to sets of coordinates defined by the functions $Neighborhood(x;y)$ and $Zone(FieldObjects, (x;y))$ for focal map cubic and zonal map cubic operators respectively (("Aggregation functions as Map/Map Cubic Algebra functions " requirement of Table 1)).

Definition 8. Aggregation of the Field attribute on non-field hierarchies using Map Algebra and Cubic Map algebra functions
Let $\mathcal{F}$ be a field attribute, and $f_1 \ldots f_n$ functions of the domain of $\mathcal{F}$ with g as field support (without loss of generality, we suppose the $f_1 \ldots f_n$ have the same field support)

$$f_1 : g \rightarrow dom_{\mathcal{F}} \ldots f_n : g \rightarrow dom_{\mathcal{F}}, \text{ and } f_1 \ldots f_n \in dom(\mathcal{F}).$$

Let $O_{\mathcal{A}}$ be an alphanumeric aggregation

Then, the aggregation of $\mathcal{F}$ is defined by means of a function $O_{\mathcal{F}}$ that takes as input $f_1 \ldots f_n$, and that returns a function f_{1n} defined on g and having values in $dom_{\mathcal{F}}$ ($f_{1n} : g \rightarrow dom_{\mathcal{F}}$) ($f_{1n} = O_{\mathcal{F}}(f_1 \ldots f_n)$) such that:

- *using Local operator:*
  $f_{1n}(x;y) = O_{\mathcal{A}}(f_1(x;y), \ldots, f_n(x;y))$ *for each point* $(x;y)$ *of* g

- *using Cubic Focal operator:*
  $f_{1n}(x;y) = O_{\mathcal{A}}(f_1(Neighborhood(x;y)) \ldots f_n(Neighborhood(x;y)))$ *for each point* $(x;y)$ *of* g *where:*
  $Neighborhood(x;y)$ *is a function that returns the neighbourhood points of* $(x;y)$,

- *using Cubic Zonal operator:*
  $f_{1n}(x;y) = O_{\mathcal{A}}(f_1(Zone(FieldObjects,(x;y)) \ldots f_n(Zone(FieldObjects,(x;y)))$ *for each point* $(x;y)$ *of* g *where:*
  $Zone(FieldObjects, (x;y))$ *is a function that takes as input a set of Field Objects and a point, and it returns the neighbourhood points of (x;y)that belong to the zone indentified by the FieldObjects on this point.*

Definition 9. built non-field (Figure 7)

Let $tb_{f1}, \ldots tb_{fk}$ and t_{nf} instances of the field object structure $S_e = \langle geom, field, [a_1, \ldots a_m] \rangle$
Let ONF, called non-field aggregation mode, a set of aggregation functions:

- *$O_{\mathcal{G}}$ the spatial aggregation for geom*
- *$O_1 \ldots O_m$ the alphanumeric aggregations for $a_1, \ldots a_m$*
- *$O_{\mathcal{F}}$ the Map Algebra/Map Cubic Algebra aggregation for field*

We say that t_{nf} is built non-field from $tb_{f1}, \ldots tb_{fk}$ using ONF if:

- $t_{nf}.geom = O_{\mathcal{G}}(tb_{f1}.geom, \ldots, tb_{fk}.geom)$
- $\forall i \in [1, \ldots m]\ t_{nf}.a_i = O_i(tb_{f1}.a_i, \ldots, tb_{fk}.a_i)$
- $t_{nf}.field = O_{\mathcal{F}}(tb_{f1}.field, \ldots, tb_{fk}.field)$

Example 6

Let an instance of S_{earthq} $t_1 = \langle p_1, f_1 \rangle$ where p_1 is a geometry and f_1 represents the intensity of the earthquake on the 9-1998 in Lombardia. It is defined on each point of p_1 with values in *R*. For example $f_1(x;y) = 10$ (Figure 6a). To aggregate the field attribute intensity on the temporal dimension, we use a cubic focal operator AVG. Therefore the result of the aggregation of f_1 and f_2 on (x;y) by taking into account neighbours of (x;y) is $f_3(x;y) = ((13*4+10)+(11*4+12))/10 = 11.7$ (we suppose that the values of neighbourhood points of (x;y) of t_1 and t_2 are 10 and 12 respectively) (Figure 6a). We suppose that we apply the geometric union for geometry.

Then, t_3 is built non-field from t_1 and t_2. t_3 is an aggregated measure of the cuboid defined by the *century* level of the temporal dimension (see Table 3).

Table 3. Instances of the cuboid defined by the *century* level of the temporal dimension

Region	Century	Earthq
Lombardia	900	t_3
Piemonte	900	t_6

Then, as a field measures is mapped also on spatial dimensions, then a particular aggregation must be provided taking into account the field hierarchy in order to allow the visualization of field measures at different resolutions or scales.

3.2.2.1.2 Aggregation on the field hierarchy

The aggregated measures of a cuboid defined by coarser spatial levels are the aggregation of the (aggregated) measures of the cuboids defined by non-spatial levels.

Definition 10. Built field (Figure 7)

Let $t^1_{nf},\ldots, t^v_{nf}$ and t_f be instances of the field object structure $S_e \langle geom, field, [a_1, \ldots a_m]\rangle$. Let $O\mathcal{F}$, called field aggregation mode, a set of aggregation functions:

- *O_G the spatial aggregation for geom*
- *$O_1 \ldots O_m$ the alphanumeric aggregations for $a_1, \ldots a_m$*
- *$O_{\mathcal{A}}$ the alphanumeric aggregation for field*

We say that t_f is built field from $t^1_{nf},\ldots, t^v_{nf}$ using $O\mathcal{F}$ if:

- *$t_f.geom = O_G\ (t^1_{nf}.geom,\ldots, t^v_{nf}.geom)$*
- *$\forall i \in [1,\ldots m]\ t_f.a_i. = O_G\ (t^1_{nf}.\ a_i,\ldots, t^v_{nf}.\ a_i)$*
- *$t_f.field\ (x;\ y) = O_{\mathcal{A}}\ (f^1(x^1;y^1),\ \ldots, f^m(x^m;y^m))$ where $f^1,\ \ldots, f^m$ belong to $t^1_{nf}.field, \ldots\ t^v_{nf}.field$, for each point $(x;y)$ of the field support of field*

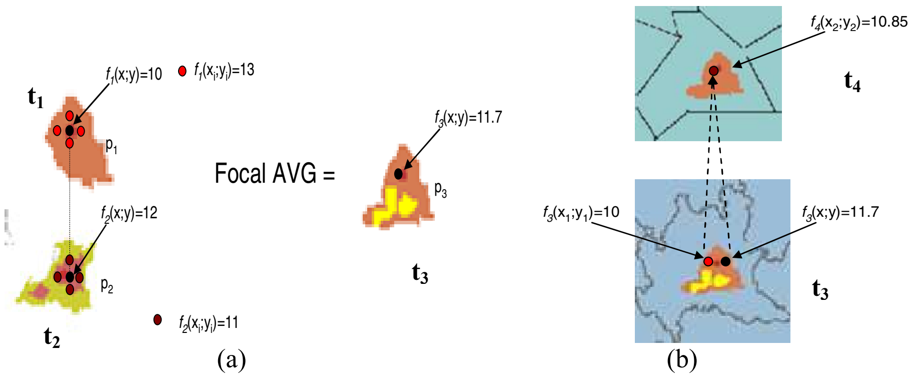


Fig. 6. a) Aggregation on the "intensity" field attribute on the temporal dimension, b) Aggregation on the "intensity" field attribute on the field hierarchy

Example 7

In order to visualize the measure at different resolutions, we aggregate on the Field Hierarchy H_{regres} applying the average. Then $f_4(x;y) = AVG(leavesFieldSupport(H_{deptres}, (x_2;y_2))) = AVG(f_3(x;y), f_3(x_1;y_1)) = (10+11.7)/2 = 10.85$ (Figure 6b). *Then, t_4 is built field from t_3.* t_4 is an aggregated measure of the cuboid defined by the *century* and *regres* levels (see Table 4).

Table 4. Instance of FC_{earthq}

Regres (region at scale 1:1.000)	Century	Earthq
Lombardia	900	t_4
Piemonte	900	t_5

3.2.2.2 Cuboids of Field Data

Once described how the measures of the different cuboids are related by aggregation functions, in this section we formalize the concept of cuboid. In particular, a cuboid schema, noted *Field View Structure*, is composed by a set of levels, a non-field aggregation mode, and a field aggregation mode. An *instance of a field view structure* is a set of tuples composed of a member for each level and a (aggregated) field measure value. The aggregated field measure value on the spatial dimension (t_f) is obtained aggregating measures (t^1_{nf} …, t^v_{nf}) obtained after the aggregation on the non-spatial dimensions of detailed measures (tb_{f1} …, tb_{fk}) as shown on Figure 7.

Definition 11. Field View

A Field View Structure $\mathcal{V}_v$ is a tuple $\langle \mathcal{FC}_c, \mathcal{L}, \mathcal{ONF}, \mathcal{OF}\rangle$ where:

- $\mathcal{FC}_c=\langle \mathcal{H}_1,\ldots\mathcal{H}_n, \mathcal{F}ieldObject\rangle$ *is a Field Cube Structure* (Spatio-multidimensional model schema)
- $\mathcal{L}$ *is a tuple* $\langle \mathcal{S}_1,\ldots\mathcal{S}_n\rangle$ *where* $\forall i \in [1,\ldots n]$ $\mathcal{S}_i$ *is a level of* $\mathcal{H}_i$ (Levels that define the cuboid)
- $\mathcal{ONF}$ *is a non field aggregation mode* (Aggregation functions used on non-spatial dimensions)
- $\mathcal{OF}$ *is a field aggregation mode* (Aggregation functions used on the spatial dimension)

An Instance of a field view Structure is a set of tuples $\{\langle t_1,\ldots t_n, t_f\rangle\}$ *where:*

- $\forall i \in [1,\ldots n]$ t_i *is an instance of* $\mathcal{S}_i$ (Dimensions members)
- t_f *is:* ((aggregated) field measure on spatial dimension, Figure 7 - see Table 4 for an example)
  - *an instance of* $\mathcal{F}ieldObject$
  - *built field from* $t^1_{nf},\ldots, t^v_{nf}$ *using* $\mathcal{OF}$ *where* ((aggregated) field measures on non-spatial dimensions, Figure 7 - see Table 3 for an example)*:*
    - $t_f.field\ (x;y)= \mathcal{OF}.O_{\mathcal{A}}\ (f^1(x^1;y^1), \ldots, f^m(x^m;y^m))$ *for each point* $(x;y)$ *of its field support where:*
      - $(x^1;y^1), \ldots, (x^m;y^m)$ *belong to leafsFieldSupport*$(\mathcal{H}_1, (x;y))$
      - $f^1, \ldots, f^m$ *belong to* $t^1_{nf}.field,\ldots, t^v_{nf}.field$
  - *Each* t^j_{nf} *is built non field from* $tb^j_{f1}, \ldots, tb^j_{fk}$ *using* $\mathcal{ONF}$ *where* (Non aggregated field measures, Figure 7 - see Table 2 for an example)*:*
    - $tb^j_{f1}, \ldots, tb^j_{fk}$ *are the measure values of the tuples of* $I(\mathcal{FC}_c)$ $\langle tb^j_1, tb^1_2\ldots tb^1_n, tb^j_{f1}\rangle, \ldots,\langle tb^j_1, tb^k_2\ldots tb^k_n, tb^j_{fk}\rangle$ *where:*
      - $\forall i \in [2,\ldots n]$ $tb^1_i \ldots tb^k_i = leafs(\mathcal{H}_i, t_i)$
      - tb^j_1 *belongs to* $leafs(\mathcal{H}_1, t_1)$

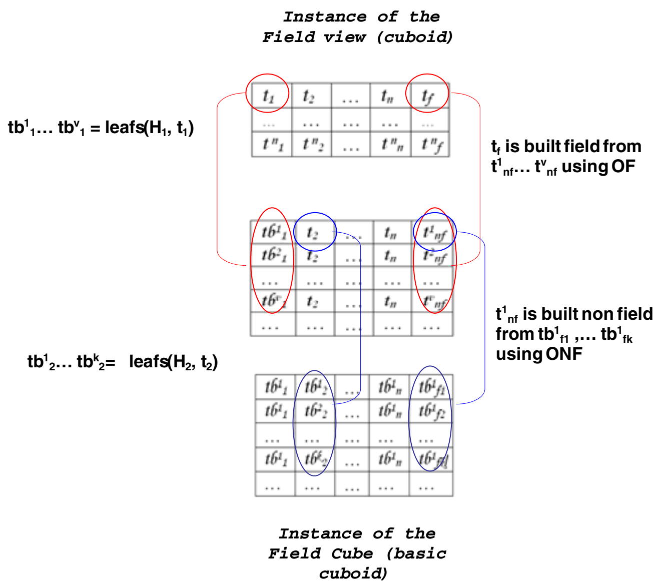

Instance of the Field view (cuboid)
tb¹₁… tbᵛ₁ = leafs(H₁, t₁)
t_f is built field from t¹_nf… tᵛ_nf using OF
t¹_nf is built non field from tb¹_f1 ,… tb¹_fk using ONF
tb¹₂… tbᵏ₂= leafs(H₂, t₂)
Instance of the Field Cube (basic cuboid)

Fig. 7. Instance of a Field View Structure

Example 8
The Field View Structure representing earthquakes per region at the scale 1:10000 and per century is $V_{\text{earthq}} = \langle FC_{earth}, \langle S_{century}, S_{regres}\rangle, \langle Union, Focal\text{-}Avg\rangle, \langle Union, Avg\rangle\rangle$. Table 4 shows its instance.

4 Conclusion and Future Work

Integration of spatial data into multidimensional models leads to the concept of SOLAP. SOLAP models exploit the discrete representation of spatial data. Few works integrate continuous field data into dimensions and measures. In this paper, motivated by the relevance of a formal representation of SOLAP data, we provide a multidimensional model that considers field data independently form their implementation, as measures and dimensions. In particular we provide a unique data model for vector and field data (Geographic and Field Objects). We provide a formal representation of the spatio-multidimensional model schema (Field Cube: Field Hierarchy and Field Measures) and the associated hypercube's cuboids (Field View).

Actually, we are working on the formal definition of SOLAP operators that allows the navigation between the cuboids (roll-up/drill-down), and slicing the cuboids (slice). We plan to work on the implementation of the model in a ROLAP architecture. This implies the definition of: (i) query languages for OLAP server [18] for field data [12], (ii) indexes [20] and pre-aggregation techniques [19] for spatial data warehouses using field dimensions and measures, and (iii) interactive field maps [17] for SOLAP clients.

References

[1] Ahmed, T., Miquel, M.: Multidimensional Structures Dedicated to Continuous Spatio-temporal Phenomena. In: Jackson, M., Nelson, D., Stirk, S. (eds.) BNCOD 2005. LNCS, vol. 3567, pp. 29–40. Springer, Heidelberg (2005)

[2] Bédard, Y., Han, J.: Fundamentals of Spatial Data Warehousing for Geographic Knowledge Discovery. In: Geographic Data Mining and Knowledge Discovery. Taylor & Francis, New York (2009)

[3] Bimonte, S., Gensel, J., Bertolotto, M.: Enriching Spatial OLAP with Map Generalization: a Conceptual Multidimensional Model. In: IEEE International Workshop on Spatial and Spatiotemporal Data Mining, pp. 332–334. IEEE CS Press, Los Alamitos (2008)

[4] Bimonte, S., Tchounikine, A., Miquel, M., Pinet, F.: When Spatial Analysis Meets OLAP: Multidimensional Model and Operators. International Journal of DataWarehousing and Mining (to appear)

[5] Câmara, G., De Freitas, U., Cordeiro, J.: Towards an algebra of geographical fields. In: Brazilian Symp. on Computer Graphics and Image Processing, Anais, Curitiba, pp. 205–212 (1994)

[6] Di Martino, S., Bimonte, S., Bertolotto, M., Ferrucci, F.: Integrating Google Earth within OLAP Tools for Multidimensional Exploration and Analysis of Spatial Data. In: ICEIS 2009. LNBIP, vol. 24, pp. 940–951. Springer, Heidelberg (2009)

[7] Franklin, C.: An Introduction to Geographic Information Systems: Linking Maps to databases. Database 15(2), 13–21 (1992)

[8] Gutierrez, A., Baumann, P.: Modeling Fundamental Geo-Raster Operations with Array Algebra. In: IEEE International Workshop on Spatial and Spatiotemporal Data Mining, pp. 607–612. IEEE CS Press, Los Alamitos (2007)

[9] Harinarayan, V., Rajaraman, A., Ullman, J.D.: Implementing Data Cubes Efficiently. In: ACM SIGMOD International Conference on Management of Data, pp. 205–216. ACM Press, New York (1996)

[10] Kemp, K.: Environmental Modeling with GIS: A Strategy for Dealing with Spatial Continuity. Technical Report 93-3. National Center for Geographic Information and Analysis, University of California, Santa Barbara, USA (1993)

[11] Kimball, R.: The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Data Warehouses. John Wiley & Sons, New York (1996)

[12] Laurini, R., Gordillo, S.: Field Orientation for Continuous Spatio-temporal Phenomena. In: International Workshop on Emerging Technologies for Geo-based Applicatons, pp. 77–101. Swiss Federal Institute of Technology, Lausanne (2000)

[13] Ledoux, H., Gold, C.M.: A Voronoi-based Map Algebra. In: 12th International Symp. on Spatial Data Handling, pp. 117–131. Springer, Heidelberg (2006)

[14] Malinowski, E., Zimányi, E.: Advanced Data Warehouse Design From Conventional to Spatial and Temporal Applications. Springer, Heidelberg (2008)

[15] McHugh, R.: Intégration De La Structure Matricielle Dans Les Cubes Spatiaux. Université Laval (2008)

[16] Mennis, J., Viger, R., Tomlin, C.D.: Cubic Map Algebra functions for spatio-temporal analysis. Cartography and Geographic Information Systems 30(1), 17–30 (2005)

[17] Plumejeaud, C., Vincent, J., Grasland, C., Bimonte, S., Mathian, H., Guelton, S., Boulier, J., Gensel, J.: HyperSmooth, a system for Interactive Spatial Analysis via Potential Maps. In: Bertolotto, M., Ray, C., Li, X. (eds.) W2GIS 2008. LNCS, vol. 5373, pp. 4–16. Springer, Heidelberg (2008)

[18] Silva, J., Castro Vera, A.S., Oliveira, A.G., Fidalgo, R., Salgado, A.C., Times, V.C.: Querying geographical data warehouses with GeoMDQL. In: Brazilian Symposium on Databases, pp. 223–237 (2007)

[19] Stefanovic, N., Han, J., Koperski, K.: Object-Based Selective Materialization for Efficient Implementation of Spatial Data Cubes. IEEE Transactions on Knowledge and Data Engineering 12(6), 938–958 (2000)

[20] Tao, Y., Papadias, D.: Historical spatio-temporal aggregation. ACM Trans. Inf. Syst. 23(1), 61–102 (2005)

[21] Timpf, S., Frank, A.U.: Using hierarchical spatial data structures for hierarchical spatial reasoning. In: Frank, A.U. (ed.) COSIT 1997. LNCS, vol. 1329, pp. 69–83. Springer, Heidelberg (1997)

[22] Tomlin, C.D.: Geographic Information Systems and Cartographic Modeling. Prentice Hall, Englewood Cliffs (1990)

[23] Vaisman, A., Zimányi, E.: A multidimensional model representing continuous fields in spatial data warehouses. In: 17th ACM SIGSPATIAL International Symp. on Advances in Geographic Information Systems, pp. 168–177. ACM Press, New York (2009)

Exploiting the Semantics of Location Granules in Location-Dependent Queries

Carlos Bobed, Sergio Ilarri, and Eduardo Mena

IIS Department
University of Zaragoza
50018 Zaragoza, Spain
{cbobed,silarri,emena}@unizar.es

Abstract. The need for location-based services has motivated an important research effort in the efficient processing of location-dependent queries. Most of the existing approaches only deal with locations at maximum precision (e.g., GPS coordinates). However, due to imprecision or expressivity requirements, there are situations in which locations must be handled at different granularity levels (e.g., neighborhoods, cities, states, etc.). Indeed, whenever a set of locations are represented together as a *granule*, a meaning is implicitly given to the set. So, the use of different granularities brings different semantics to the location data.

In this paper, we propose the use of *semantic location granules* to enhance the expressivity of location-dependent queries. This is done by exploiting the semantic information that is asserted about different granularity levels. This information could be, for example, the cost incurred by a moving object to traverse a spatial area or a requirement to traverse a connection (e.g., need of a visa or passport). In particular, we propose: 1) an ontological model for describing the semantics inherent to location granules; 2) an upper-level ontology that can be extended and adapted to different scenarios; and 3) the use of a reasoner to exploit the semantics expressed in the ontologies, to make it possible to add new query constraints and so extend the expressivity of the queries.

1 Introduction

Nowadays, the interest in mobile computing has grown due to the ever-increasing use of mobile devices and their pervasiveness. The computing capabilities of mobile devices are also growing and users are demanding data access anywhere and at anytime. This has motivated, in the mobile computing field, an intensive research in *Location-Based Services* (LBS) [22]. These services provide value-added by considering the locations of the mobile users to offer customized information.

Processing location-dependent queries has been the subject of intense research [20], as it is a major building block of location-based services. Existing works on location-dependent query processing implicitly assume GPS locations for the objects in a scenario (e.g., [4,10]). However, precise locations may be unavailable or even be inconvenient for the user. In those scenarios, it is useful to define the concept of *location granule* (similar to the concept of *place* in [14]) as a set of physical locations. Some examples of location granules could be: freeways, buildings, offices in a building, etc.

The use of location granules to enhance the expressivity of location-dependent queries was first proposed in [18]. By using location granules in query constraints, the user is able to express queries and retrieve results according to the needed resolution. As described in that work, the use of location granules can have an impact on: 1) the presentation of results (location granules can be represented by using graphics, text, sounds, etc., depending on the requirements of the user), 2) the expressivity of the queries (the user expresses the queries according to his/her location terminology, and therefore the answers to those queries will depend on the interpretation of location granules), and 3) the performance of the query processing (the location tracking overload is alleviated when coarse location granules, instead of precise GPS locations, are used). In [17], the implications of using location granules in inside and nearest neighbour constraints were studied, and the granule-based query processing was also extended to deal with uncertainty, which led to consider *probabilistic granule-based inside queries* and *probabilistic granule-based nearest neighbour queries.*

However, thus far, the impact of location granules regarding location-dependent queries has only been studied from the point of view of query processing, leaving aside concerns about the semantics of location granules. Looking at the big picture, whenever a set of locations is represented as a location granule, we are assigning it an implicit meaning (e.g., cities, neighbourhoods, etc.). Even when the use of location granules is forced by the resolution given by the location service used, an implicit meaning arises (e.g., mobile phone cells). Making these meanings explicit would allow to further extend the expressivity of granule-based location-dependent queries. Besides, several properties and relationships could be considered once we have stated the exact interpretation of the location granules (e.g., when considering countries, different tolls could be established for traversing different boundaries).

In this paper, we propose to study the semantic dimension of location granules, extending their definition to obtain *semantic location granules.* This approach, which is complementary to our previous works on location granules [17,18], allows to make their implicit semantics explicit and to take advantage of them to further extend the semantics of granule-based location-dependent queries. Firstly, we define the notion of *semantic location granule* and advocate the use of ontologies (which offer a formal, explicit specification of a shared conceptualization [11]) to represent them, by making explicit their properties, their relationships, and other logical rules that capture their semantics. Secondly, we analyze the most basic relationships that exist between location granules and propose a base ontology that can be extended and adapted to different scenarios. Thirdly, by using ontologies, and with the help of a reasoner [1], we propose an approach that allows extending dynamically the expressivity of the queries that can be processed. This extension of the expressivity is done by asserting new properties, relationships, and rules, into the granule ontology used. Using the inference capabilities of the reasoner, we make it possible to use this newly asserted knowledge to build new query constraints with the semantics explicitly stated in the ontology.

The structure of the rest of the paper is as follows. In Section 2, we present two running examples that will accompany the explanations along the paper. In Section 3, the definition of *semantic location granules* is provided. In Section 4, we describe the base ontology that we propose. In Section 5, we focus on studying the impact of the added semantics in the processing of *inside* constraints. In Section 6, we present some related works. Finally, some conclusions and plans for future work appear in Section 7.

2 The Importance of Adding Semantics

In this section, we briefly overview the basics of the use of location granules in location-dependent queries, and then we show a couple of motivating examples.

2.1 Location-Dependent Query Processing with Location Granules

For illustrative purposes, we use an SQL-like syntax along the paper to express the queries and constraints, which allows to emphasize the use of location granules and state the queries concisely. The structure of location-dependent queries is:

```
SELECT projections
  FROM sets-of-objects
 WHERE boolean-conditions
```

where *sets-of-objects* is a list of object classes that identify the kind of objects interesting for the query, *boolean-conditions* is a boolean expression that selects objects from those included in *sets-of-objects* by restricting their attribute values and/or demanding the satisfaction of certain *location-dependent constraints*, and *projections* is the list of attributes or location granules that must be retrieved from the selected objects. Specifications of the use of granules can appear in the SELECT and/or in the WHERE clause of a query, depending on whether those location granules must be used for the visualization of results or for the processing of constraints, respectively. If no location granules are specified, GPS locations are assumed.

In Figure 1, an example of how to process a granule-based inside constraint is shown. The general syntax of an *inside* constraint is *inside(r, obj, target)*, which retrieves the objects of a certain class *target* (such objects are called *target objects* and their class the *target class*) within a specific distance *r* (which is called the *relevant radius*) of a certain moving object *obj* (that is called the *reference object* of the constraint). Thus, for example, the constraint *inside(130 miles, gr(province, car38), gr(province, Car))* retrieves the cars in provinces within 130 miles of the province where car38 is located. To process that constraint: 1) the granule where the reference object (car38) is located is obtained and a buffering

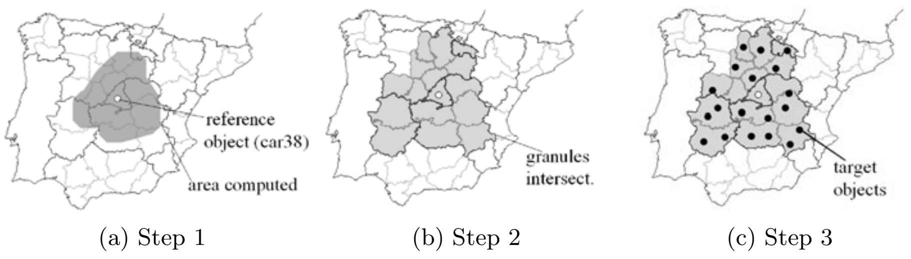


Fig. 1. Inside with granules for the reference object and the target class: steps

operation is performed to obtain the surrounding area within 130 miles; 2) the granules that intersect that area are retrieved; and 3) the objects of the class *Car* that are inside those relevant granules are retrieved. For further details on how to process granule-based constrains we refer the reader to [17,18].

2.2 Motivating Examples

In this section we present two examples that show that, although the semantics of the queries are extended by the use of granules, there is still a need for richer semantics. We will re-visit these examples in Section 5.3, once we have proposed a solution to represent the missing semantics.

Traffic Monitoring Example. The first example is a traffic monitoring application. In spite of using a traffic application as motivating example, we want to remark that our approach does not focus on query processing on road networks, as opposed to works such as [16,24]. Thus, neither the trajectories of the moving objects in our approach are restricted to a fixed network nor the location granules are limited to represent roads. In fact, the granules we have modelled for the example can be considered as coarser representations of the motorways, which might include some fragments of surrounding roads. As we are not limited to work with road segments (arbitrary regions can be used to define the location granules), we can obtain also the objects that are likely to use the motorways without having to specify all the surrounding roads. Thus, we consider works that model objects moving on road networks complementary to our approach, as they could adopt our approach to add a separate layer to take into account semantic aspects that otherwise would be lost.

At a particular moment in time, let us suppose that we want an overview of the evolution of the traffic density that the A2 motorway is supporting, along with its surroundings, to be able to foresee possible traffic jams in that entry of Madrid. We have modelled the motorways entering Madrid as shown in Figure 2. A query with an inside constraint such as:

SELECT *COUNT(Car.id)*
FROM *Car*
WHERE *inside(50 miles,* $A2_1$*, Car)*

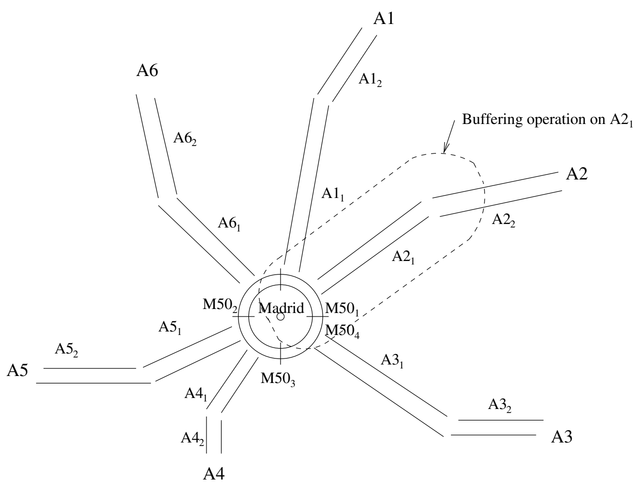


Fig. 2. Location granule map: motorways entering Madrid

would seem perfect to obtain the number of cars going to Madrid using the A2 motorway; however, its semantics are not accurate enough because the buffering operation (see Section 2.1, and particularly the step 1 in the example of Figure 1) over the $A2_1$ granule could retrieve also cars that are going through the A1 and A3 motorways, which would introduce noise in the answer. This problem is not easy to solve. For example, using granules for the target objects of the query:

```
SELECT COUNT(Car.id)
  FROM Car
 WHERE inside(50 miles, A2₁, gr(Motorways, Car))
```

does not resolve our problem as, despite retrieving the cars in $A2_1$ and $A2_2$, the system would also retrieve all the cars in $A1_1$, $A3_1$, $M50_1$, and $M50_4$.

Taxi Monitoring Example. Let us move to another example: a taxi cab monitoring application. Leaving aside the distance, the cost of taxi services usually depends on different aspects of the origin and target destination. Imagine a person that has already landed in the airport of Barajas (Madrid), and that s/he wants to call a taxi to go to the suburbs. S/he may also know that a taxi that might be closer in terms of physical distance could be more expensive than another that is farther, due to special taxes that are applied when taxis have to move into other zones. Using location granules in the queries allows to consider the taxis according to the different zones, but not the additional costs that the taxi driver is going to charge to the user.

3 Semantic Location Granules

In previous works [17,18], we proposed a definition of *location granule* from a physical point of view (see Definition 1).

Definition 1. *A* location granule *is a set of one or more geographic areas which represent a set of GPS locations under a common name.*

For example, the set of locations that belong to *Madrid* would be a location granule. In this section, we bring this definition to a higher abstraction level to capture the semantic dimension of location granules. Let us analyse what we do when forming a location granule: we group a set of locations and give them a name. This way we are also giving them a meaning (in this case, *Madrid* is a city, the capital of Spain). The grouped locations become a new different entity as a whole, which could be related to other location granules in different ways besides spatial relations.

To formalize these notions, we advocate the use of ontologies [11]. Ontologies offer a formal, explicit specification of a shared conceptualization. Mathematically, an ontology can be seen as a tuple $< C, R, I, Rl >$, where C would be the set of concepts and their definitions, R the set of roles (relationships between concepts), I the set of instances that belong to the different concepts (the ontology population), and Rl the set of logical rules that are to be considered[1]. Regarding location granules and their properties:

– Focusing on the semantic dimension of location granules, the bare notion of location granule is itself a concept and the concrete location granules would be its instances. The set composed by the basic type of *Granule* and the possible definitions of specialized types, along with the set of instances of those location granule definitions, map directly to the C and I sets, respectively. For example, *Madrid* would be an instance of the *city* concept.
– The different characteristics of the granules would be the attributes (*datatype properties*) of the concept. They might be or not geographical properties. For example, when considering cities, we could want to know their extension but also other characteristics such as possible fees that would be charged if we visit them (for example, in Mallorca, Spain, a special EcoTax fee is charged to travelers in order to fund environment protection). Moreover, the possible relationships between location granules would be the roles of an ontology (*object properties*). For example, considering countries, there could be special visa requirements to travel from one country to another. Both sets of attributes and relationships map directly to the R set.
– Finally, apart from the definitions of the granules, we can provide rules that allow to represent dynamic facts and extend the reasoning about the granules. This set of rules would directly map to the Rl set.

[1] In OWL [13], the reference language for ontologies in the Web, *Concepts* correspond to *Classes*, *Roles* correspond to *ObjectProperties* and *DataTypeProperties*, and *Individuals* to the homonym elements.

So, the use of ontologies fits perfectly the definition of location granule and its semantics. Making the semantics of location granules explicit turns them into *semantic location granules* (see Definition 2).

Definition 2. *A* semantic location granule *is a location granule with well-defined semantics, i.e., explicitly stated.*

The most important operator for a location granule is *inGr* (short for *inGranule*), which returns a boolean indicating whether a certain GPS location is within the granule.

Location granules will not be used in isolation in an application. Instead, they participate in groups of locations granules that provide an interpretation of the location space, conforming semantic layers or *semantic granule maps* (see Definition 3).

Definition 3. *A* semantic granule map *is a set of semantic granules identified by a common name. It provides the global semantics of the location granules that participate in it.*

Following with the previous example, *Madrid* is a *cityGranule*, but it is assigned different global semantics if it participates in an application as being part of the *citiesOfSpain* or *capitalsOfTheWorld* sets of granules. Different semantic granule maps could be defined over the same geographic area.

The most important operators for location granule maps are *getGrs* (*getGranules*) and *getGrsObj* (*getGranulesObject*), which return the subset of granules that contain a specified GPS location or object, respectively. In the rest of the paper, when referring to location granules and granule maps, we mean semantic ones, and, for brevity, we will use *gr* instead of *getGrs*.

4 Modeling Location Granules with Ontologies

In the previous section, we have seen the definition of *semantic location granules*. In this section, we analyze the most basic properties of the granules and propose a base ontology for representing them and their associated granule maps.

4.1 Base Ontology for Location Granules

This ontology (see Figure 3) is not meant to be complete, but a starting point to be extended and adapted to particular scenarios. The most basic properties that we identify are the following:

– *Contains*: it represents the physical inclusion of a granule inside another. For example, the granule *Spain* (the country) contains, among others, the granules *Madrid* and *Barcelona* (the cities). This property permits to establish a spatial hierarchy to organize the granule instances. *IsContained* is its inverse property.

– *Groups*: it represents the relationship that there exists between a granule map and the granules that make it up. For example, the granule map *Countries* would group the granules *Spain*, *France*, etc. *Participates* is its inverse property. Note that a granule can participate in several granule maps.
– *Encapsulates*: it allows to establish hierarchies between granule maps according to the granularity level. For example, the granule map *provincesOfSpain* could encapsulate *citiesOfSpain*. *IsEncapsulated* is its inverse property.

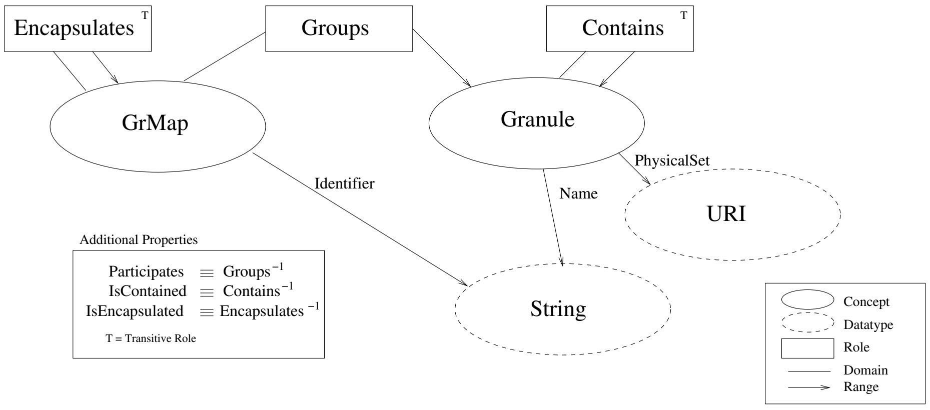


Fig. 3. Base ontology that can be dynamically extended

The rest of the properties are used to identify the granules (*Name*) and the granule maps (*Identifier*), and to associate a granule to one or several sets of physical coordinates (*PhysicalSet*). The physical coordinates are also accessible at the semantic level to allow including statements about them in the asserted knowledge and, therefore, to allow the system to perform spatial reasoning (for example, using RCC [12,21]).

Although this basic ontology may seem too simple, direct benefits can be obtained out of it. For example, even without any additional extension in the semantics, the presentation of results of the queries can be enhanced by exploiting the inclusion relationships to offer different views of the same answer set. Besides, we wanted to keep the model as simple as possible to make it easier to adapt it to the desired semantics.

4.2 Extending the Base Ontology

The proposed ontology can be loaded and handled by a Description Logics (DL) reasoner [1]. This allows making new knowledge available to the system dynamically by asserting it into the ontology. This task is meant to be performed by the final application developer or an advanced user. To extend the semantics we could specialize the existing concepts or add new properties. To illustrate it, let us get back to the examples. In the taxi monitoring application, if we want to consider zones with no additional initial cost, we could assert the following:

$$\begin{aligned}
&1) InitialCost : datatypeProperty\\
&\quad domain(InitialCost) = Granule\\
&\quad range(InitialCost) = float\\
&\quad functional\\
&2) InitialCostFreeZone := Granule\ and\ (InitialCost = 0)
\end{aligned}$$

Where *functional* indicates that the property can only hold a single value for a given individual. Let us focus now on the traffic monitoring example. We want to add a new role to the ontology to express the existence of a direct connection from one granule to another (an object can move from one granule to another directly without going through a different granule[2]). Of course, we also need to assert information about how the instances are related, but this process can be automated by using propagation rules; so, in the first example, we would assert:

$$\begin{aligned}
&1) DirectlyConnected : role\\
&\quad domain(DirectlyConnected) = Granule\\
&\quad range(DirectlyConnected) = Granule\\
&\quad symmetric, reflexive\\
&2) DirectlyConnected(A2_1, A2_2),\ DirectlyConnected(A2_1, M50_1)\\
&\quad DirectlyConnected(A3_1, A3_2),\ DirectlyConnected(A3_1, M50_4)\\
&\quad \ldots
\end{aligned}$$

Along with the following Horn rule[3]:

$$\begin{aligned}
&IsContained(?x, ?y) \wedge IsContained(?u, ?v) \wedge\\
&\wedge Participates(?y, ?z) \wedge Participates(?v, ?z) \wedge\\
&\qquad \wedge DirectlyConnected(?x, ?u) \longrightarrow DirectlyConnected(?y, ?v)
\end{aligned}$$

The assertion of the rule allows to spread automatically (thanks to the use of a DL reasoner) the knowledge asserted, which makes it usable by other granule maps that might be using the upper-level granules; for instance, a granule A2 which contains the A21 and A22 segments would be automatically detected as directly connected to M50, as one of its inner granules is directly connected to one of the inner granules of M50. The inclusion of both *Participates* clauses in the Horn rule is to control that different granularities are not mixed.

[2] Note that this property is different from just sharing some edges in the boundary, as the existence of a shared boundary between granules in fact does not assure the possibility to traverse it.

[3] It can be expressed in SWRL [15], an extension of OWL to support rule-based inferences.

5 Extending the Semantics of Inside Queries

In this section, we explain how to take advantage of the explicit semantics of the granules to enhance the expressivity of granule-based inside constraints. Firstly, we extend the operators defined for granules to be able to take into account the new semantics. Then, we analyze how this affects the definitions of granule-based inside constraints. Finally, the examples in Section 2 are reconsidered with the new available semantics.

5.1 Extending the GetGranules Operator with a Holds Function

Before extending the *gr* operator (*getGranules*, see Section 3), we have to introduce the *holds* function. *Holds* takes as input a semantic location granule (*slg*) and an expression that it has to satisfy in order to be considered relevant for the query. The allowed operators in the expression are *And*, *Or* and *Not*. The atoms used in the expression are the names of the concepts or the properties of the (possibly extended) ontology, so *holds* has to be defined separately for evaluating concepts or properties between individuals or sets of individuals:

$$holds(slg, concept) = true \Leftrightarrow slg \in concept$$
$$holds(slg, property, slg') = true \Leftrightarrow property(slg, slg') \in property$$
$$holds(slg, property, \{slg_i\}) = true \Leftrightarrow \exists slg' \in \{slg_i\} \mid holds(slg, property, slg')$$

These definitions assume a Closed World scenario[4]. When dealing with ontologies and reasoners, we have to bear in mind that they usually assume Open World scenarios. This implies that the reasoner cannot infer anything which has not been directly asserted or inferred from the previously asserted axioms. This affects directly the definition and evaluation of the *holds* function. Assuming an Open World scenario, *holds*[5] would be defined as:

$$holds(slg, concept) \begin{cases} slg \notin concept \Rightarrow false \\ otherwise \Rightarrow true \end{cases}$$
$$holds(slg, property, slg') \begin{cases} property(slg, slg') \notin property \Rightarrow false \\ otherwise \Rightarrow true \end{cases}$$
$$holds(slg, property, \{slg_i\}) \begin{cases} \forall slg' \in \{slg_i\}, property(slg, slg') \notin property \Rightarrow false \\ otherwise \Rightarrow true \end{cases}$$

In this context, it seems to be more useful to have a Closed World instead of an Open World semantics as, in fact, the facts to be asserted can be controlled. So, assuming Closed World, we redefine the *gr* operator as a three argument function with an optional fourth argument:

$$gr(m, obj, cond) = \{slg \mid participates(slg, m) \wedge inGr(slg, obj.loc) \wedge \wedge holds(slg, cond)\}$$
$$gr(m, obj, cond, \{slg_i\}) = \{slg \mid participates(slg, m) \wedge inGr(slg, obj.loc) \wedge \wedge holds(slg, cond, \{slg_i\})\}$$

[4] If the Reasoner cannot retrieve the exact fact, *holds* is evaluated to false.

[5] In an Open World scenario, whenever in doubt, *holds* is evaluated to true.

being m an instance of granule map, *obj* an object, and *cond* a list of conditions that the granules comprising the answer set must fulfil.

5.2 Extended Granule-Based Inside Constraints

In this section, we focus on how the change in the *gr* operator affects the semantics of granule-based inside constraints. To ease the explanations, and without loss of generality, we consider only constraints with concepts. If properties are to be used in the constraints, the *gr* operator would also take a set of granules as fourth argument as defined above.

Inside constraint with a granule for the reference object. In this case, the corresponding *inside* constraint is now interpreted as follows:

$$\mathit{inside}(r, \mathit{gr}(\mathit{map}, \mathit{obj}, \mathit{cond}), \mathit{target}) = \{o_i \mid (o_i \in \mathit{target}) \wedge (\exists p \in \mathit{GPS} \mid \\ \mathit{inGr}(\mathit{gr}(\mathit{map}, \mathit{obj}, \mathit{cond}), p) \wedge \mathit{distance}(p, (o_i.\mathit{loc}.x, o_i.\mathit{loc}.y)) \leq r)\}$$

Why could we want to express a constraint over the granule where the reference object is? There are many reasons, e.g., the reference object might be crossing a granule that does not hold some specific condition. For example, imagine a person walking in the rain in a zone in which taxis are not allowed to enter (or they do not want to because it is a dangerous zone). It would make no sense to ask for taxis there as that granule is defined as a taxi-free zone; so, the application could automatically add this constraint and inform the user. Moreover, granules might overlap each other, so it is possible for a location to be in two granules at the same time. In this way, we could restrict which of the potential granules are to be considered. Notice that the *inGr* operator in the formula above (see Section 3) evaluates as *false* when the *gr* operator returns no granules.

Inside constraint with granules for the target objects. An *inside* constraint can include a granule map for the target class, in which case the constraint is interpreted as follows:

$$\mathit{inside}(r, \mathit{obj}, \mathit{gr}(\mathit{map}, \mathit{target}, \mathit{cond})) = \{o_i \mid (o_i \in \mathit{target}) \wedge (\exists p \in \mathit{GPS} \mid \\ \mathit{inGr}(\mathit{gr}(\mathit{map}, o_i, \mathit{cond}), p) \wedge \mathit{distance}(p, (\mathit{obj}.\mathit{loc}.x, \mathit{obj}.\mathit{loc}.y)) \leq r)\}$$

As mentioned previously, when no granule holds the condition, *gr* will return an empty list of granules and, consequently, *inGr* will be evaluated to *false.*

Inside constraint with granules for the reference and target objects. This final situation is a mixture of the two previous cases. The new *inside* constraint is interpreted as follows:

$$\mathit{inside}(r, \mathit{gr}(\mathit{map}1, \mathit{obj}, \mathit{cond}1), \mathit{gr}(\mathit{map}2, \mathit{target}, \mathit{cond}2)) = \\ \{o_i \mid (o_i \in \mathit{target}) \wedge \; (\exists p_1, p_2 \in \mathit{GPS} \mid \mathit{distance}(p_1, p_2) \leq r \wedge \\ \wedge \; \mathit{inGr}(\mathit{gr}(\mathit{map}1, \mathit{obj}, \mathit{cond}1), p_1) \wedge \; \mathit{inGr}(\mathit{gr}(\mathit{map}2, o_i, \mathit{cond}2), p_2))\}$$

Note that we can specify different sets of conditions as well as different granule maps for the reference object and the target class. This provides the user with huge flexibility to express the conditions over the granules involved in the query.

5.3 Reconsidering the Examples with Semantic Location Granules

With the newly added semantics, the user of the traffic monitoring application can now pose the exact query (with the help of a GUI):

```
SELECT COUNT(Car.id)
  FROM Car
 WHERE inside(50 miles, A2_1,
                                gr(Motorways, Car, DirectlyConnected, {A2_1}))
```

Now, the system will only retrieve the cars that are in segments directly connected to the segment whose traffic we want to monitor. On the other hand, the taxi user that wanted to save money could pose a query such as[6]:

```
SELECT Taxi.pos
  FROM Taxi
 WHERE inside(1 mile, me, gr(TaxiZones, Taxi, InitialCostFreeZone))
```

to look for taxis within one mile in zones that would not imply initial additional costs. Note that, to extend the expressivity of the queries, in the first example we are using a property and in the second one a concept definition in the corresponding constraints.

6 Related Work

Several works on spatial databases and geographic information systems deal with the problem of managing spatial data at different levels of detail (e.g., [5,9]). These works focus on the problem of dealing with different levels of detail/specification of the spatial entities, and therefore they do not consider the use of locations at different granularities to enhance the expressiveness of queries. In the same field, the use of ontologies is mainly devoted to achieve data integration and interoperability between systems [2,3,8,25].

In the field of pervasive computing, several works have emphasized the importance of adding semantics to location data [7,23], although they do not look at this problem from the perspective of a user that wants to express queries using a suitable location granularity and with the required semantics:

– In [7], according to pervasive computing criteria, a taxonomy of types of locations is given. Therefore, locations are assigned implicit meta-semantics depending on the classification.
– In [23], the authors propose a location model that supports different expressive representations for spaces, spatial relationships, and positioning systems. However, this work focuses on positioning systems, with an emphasis in sensor data fusion. Thus, for example, the location measured by a sensor may have associated a symbolic representation. On the contrary, we handle symbolic representations (location granules) at a higher level, from the point of

[6] Where "me" is interpreted as the location of the user.

view of the user (the user uses the terminology that s/he requires). Moreover, we concern about how this can enhance the expressivity of location-dependent queries on moving objects.

There are also works that have considered locations at different granularity levels but with no semantic information attached, such as [6]. However, the authors of this work focus on data representation, leaving aside the semantics that we want to add to our granularity model. Their approach is complementary to our work, as ours can be used directly on top of their model.

Finally, existing works on location-dependent query processing implicitly assume GPS locations for the objects in a scenario. As it is difficult to provide a good overview of contributions in this area in a short space, we refer the interested reader to [20]. Although some works acknowledge the importance of considering different location resolutions (e.g., [14]), the processing of classical constraints such as *inside* or *nearest* is not considered in that context. No existing proposal has considered using ontologies to extend the expressivity of the queries dynamically.

7 Conclusions and Future Work

The bare use of granules in location-dependent queries enhances their expressivity. It brings the query to the user's level and may impact not only the query semantics but also the performance and the way the results are presented to the user. In this paper, we have studied how *semantic location granules* can be used to further enhance the expressivity of this kind of queries. In particular, our proposal:

– provides a definition of *semantic location granules* within the formal framework given by ontologies.
– offers a base ontology to represent the different granularities and the relationships that might exist. This ontology is intended to serve as starting point and can be easily extended to represent the semantics of the relationships that the user might want to add to the system.
– dynamically adapts the query processing to the newly added semantics, allowing to extend the expressivity of the queries that can be processed.

Besides, we have illustrated how this approach allows us to express queries that would not be possible otherwise. As far as the authors know, there is no other proposal that considers the use of ontologies to extend dynamically the expressivity of granule-based location-dependent queries.

As future work, we plan to integrate the proposal into the distributed location-dependent query processing system LOQOMOTION [19], adapting the granule-based processing that it is currently implemented. We are also studying how to extend other constraints with our semantic approach.

Acknowledgments

This work has been supported by the CICYT project TIN2007-68091-C02-02.

References

1. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Pastel-Scheneider, P.: The Description Logic Handbook. In: Theory, Implementation and Applications. Cambridge University Press, Cambridge (2003)
2. Bittner, T., Donnelly, M., Smith, B.: A spatio-temporal ontology for geographic information integration. International Journal of Geographical Information Science 23, 765–798 (2009)
3. Cai, G.: Contextualization of geospatial database semantics for human–GIS interaction. Geoinformatica 11, 217–237 (2007)
4. Cai, Y., Hua, K.A., Cao, G., Xu, T.: Real-time processing of range-monitoring queries in heterogeneous mobile databases. IEEE Transactions on Mobile Computing 5, 931–942 (2006)
5. Camossi, E., Bertolotto, M., Bertino, E., Guerrini, G.: Issues on modeling spatial granularities. In: COSIT 2003 Workshop: Fundamental Issues in Spatial and Geographic Ontology, Ittingen, Switzerland. Springer, Heidelberg (September 2003)
6. Camossi, E., Bertolotto, M., Bertino, E., Guerrini, G.: A multigranular spatiotemporal data model. In: 11th ACM Intl. Symposium on Advances in Geographic Information Systems (GIS 2003), New Orleans, Louisiana, USA, pp. 94–101. ACM, New York (2003)
7. Dobson, S.: Leveraging the subtleties of location. In: 2005 Joint Conference on Smart Objects and Ambient Intelligence (sOc-EUSAI 2005), Grenoble, France, pp. 189–193. ACM, New York (2005)
8. Fonseca, F., Egenhofer, M., Agouris, P., Câmara, C.: Using ontologies for integrated geographic information systems. Transactions in GIS 6, 231–257 (2002)
9. Fonseca, F., Egenhofer, M., Davis, C., Câmara, G.: Semantic granularity in ontology-driven geographic information systems. Annals of Mathematics and Artificial Intelligence 36(1-2), 121–151 (2002)
10. Gedik, B., Liu, L.: Mobieyes: A distributed location monitoring service using moving location queries. IEEE Transactions on Mobile Computing 5(10), 1384–1402 (2006)
11. Gruber, T.R.: Towards principles for the design of ontologies used for knowledge sharing. In: Guarino, N., Poli, R. (eds.) Formal Ontology in Conceptual Analysis and Knowledge Representation, Deventer, The Netherlands. Kluwer Academic Publishers, Dordrecht (1993)
12. Grütter, R., Scharrenbach, T., Bauer-Messmer, B.: Improving an RCC-derived geospatial approximation by OWL axioms. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 293–306. Springer, Heidelberg (2008)
13. Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P.F., Rudolph, S.: OWL 2 web ontology language primer. W3C Recommendation (November 2009)
14. Hoareau, C., Satoh, I.: A model checking-based approach for location query processing in pervasive computing environments. In: OTM 2007 Workshops (PerSys 2007), Algarve, Portugal, November 2007, pp. 866–875. Springer, Heidelberg (2007)
15. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B., Dean, M.: SWRL: A Semantic Web Rule Language combining OWL and RuleML. W3C Member Submission (May 2004)
16. Huang, R., Peng, Z.R.: A spatiotemporal data model for dynamic transit networks. International Journal of Geographical Information Science 22, 527–545 (2008)

17. Ilarri, S., Corral, A., Bobed, C., Mena, E.: Probabilistic granule-based inside and nearest neighbor queries. In: Grundspenkis, J., Morzy, T., Vossen, G. (eds.) ADBIS 2009. LNCS, vol. 5739, pp. 103–117. Springer, Heidelberg (2009)
18. Ilarri, S., Mena, E., Bobed, C.: Processing location-dependent queries with location granules. In: OTM 2007 Workshops (PerSys 2007), Algarve, Portugal, pp. 856–866. Springer, Heidelberg (November 2007)
19. Ilarri, S., Mena, E., Illarramendi, A.: Location-dependent queries in mobile contexts: Distributed processing using mobile agents. IEEE Transactions in Mobile Computing 5(8), 1029–1043 (2006)
20. Ilarri, S., Mena, E., Illarramendi, A.: Location-dependent query processing: Where we are and where we are heading. ACM Computing Surveys 42(3), 1–73 (2010)
21. Randell, D.A., Cui, Z., Cohn, A.G.: A spatial logic based on regions and connection. In: 3rd. Intl. Conference on Knowledge Representation and Reasoning, pp. 165–176. Morgan Kauffmann, San Francisco (1992)
22. Schiller, J., Voisard, A. (eds.): Location-Based Services. Morgan Kaufmann, San Francisco (2004)
23. Stevenson, G., Ye, J., Dobson, S., Nixon, P.: LOC8: A location model and extensible framework for programming with location. IEEE Pervasive Computing 9, 28–37 (2010)
24. Vazirgiannis, M., Wolfson, O.: A spatiotemporal model and language for moving objects on road networks. In: Jensen, C.S., Schneider, M., Seeger, B., Tsotras, V.J. (eds.) SSTD 2001. LNCS, vol. 2121, pp. 20–35. Springer, Heidelberg (2001)
25. Visser, U., Stuckenschmidt, H., Schuster, G., Vögele, T.: Ontologies for geographic information processing. Computers & Geosciences 28, 103–117 (2002)

On a Fuzzy Group-By and Its Use for Fuzzy Association Rule Mining

Patrick Bosc, Olivier Pivert, and Grégory Smits

Irisa – Enssat, University of Rennes 1
Technopole Anticipa 22305 Lannion Cedex France
bosc@enssat.fr, pivert@enssat.fr, smits@irisa.fr

Abstract. Group-by is a core database operation that is used extensively in data analysis and decision support systems. In many application scenarios, it appears useful to group values according to their compliance with a certain concept instead of founding the grouping on value equality. In this paper, we propose a new SQLf construct that supports fuzzy-partition-based group-by (FGB). We show that FGB can be used to generate fuzzy summaries as well as to mine fuzzy association rules (whose head or body are bound to a specific fuzzy value) in a practical and efficient way.

1 Introduction

The relationships between databases and fuzzy sets have been studied for a long time. Among the very first works is that by V. Tahani's in the 70s [1], which was about the design of new querying capabilities for relational databases. The basic idea was to use fuzzy sets in order to devise new predicates intended for representing a graded satisfaction instead of the "all-or-nothing" behavior conveyed by Boolean conditions. If this type of issue was quite innovative at that time, it turns out that it has gained more and more acceptance for some years in the database community. Many fuzzy querying approaches have ben proposed in the last two decades, among which a fuzzy extension of the SQL language, called SQLf [2]). This language has the same general philosophy as SQL (as to querying features and syntax in particular) and offers new possibilities regarding flexible querying. The underlying principle is to introduce graduality wherever it appeared meaningful (by using fuzzy predicates in the *where* clause of a base block, but also by introducing fuzzy nesting operators, fuzzy quantifiers and so on). However, the *group-by* clause in SQLf remained unchanged with respect to SQL. As noted in [3], grouping capabilities have been extensively studied and implemented in data management systems. The standard *group-by* operator has relatively good execution time and scalability properties. However, while its semantics is simple, it is also limited because it is based only on equality, i.e., all the tuples in a group have exactly the same values of the grouping attributes. In this paper, we aim to extend this core database operation by defining a "fuzzy grouping" mechanism. The contributions of this paper are as follows:

– we introduce the fuzzy group-by (FGB) operator which extends standard group-by to allow the formation of groups based on predefined fuzzy partitions of the attribute domains rather than equality of data,

- we show how this mechanism makes it possible to perform some kind of data summarization "on demand",
- we point out the interest of the FGB operator for the purpose of fuzzy association rule mining.

The remainder of the paper is organized as follows. Section 2 recalls some basic notions of fuzzy set theory and presents the general framework of the SQLf language. Section 3 introduces the definition of the FGB operator and shows how fuzzy summaries can be obtained by means of appropriate aggregates applied to the fuzzy groups that are produced by this operator. Section 4 discusses different forms that the complementary *having* clause can take. Section 5 deals with the way FGB can be used for mining fuzzy association rules whose head or body are bound to a specific fuzzy value. Section 6 briefly tackles implementation aspects. Related work is presented in Section 7, and Section 8 gives the conclusions and some directions for future research.

2 Reminder about Fuzzy Sets and Fuzzy Queries

2.1 Basic Notions about Fuzzy Sets

Fuzzy set theory was introduced by Zadeh [4] for modeling classes or sets whose boundaries are not clear-cut. For such objects, the transition between full membership and full mismatch is gradual rather than crisp. Typical examples of such fuzzy classes are those described using adjectives of the natural language, such as *young, cheap, fast*, etc. Formally, a fuzzy set F on a referential U is characterized by a membership function $\mu_F : U \to [0, 1]$ where $\mu_F(u)$ denotes the grade of membership of u in F. In particular, $\mu_F(u) = 1$ reflects full membership of u in F, while $\mu_F(u) = 0$ expresses absolute non-membership. When $0 < \mu_F(u) < 1$, one speaks of partial membership.

Two crisp sets are of particular interest when defining a fuzzy set F:

- the core $C(F) = \{u \in U \mid \mu_F(u) = 1\}$, which gathers the *prototypes* of F,
- the support $S(F) = \{u \in U \mid \mu_F(u) > 0\}$.

In practice, the membership function associated with F is often of a trapezoidal shape. Then, F is expressed by the quadruplet (A, B, a, b) where $C(F) = [A, B]$ and $S(F) = [A - a, B + b]$, see Figure 1.

Let F and G be two fuzzy sets on the universe U, we say that $F \subseteq G$ iff $\mu_F(u) \leq \mu_G(u)$, $\forall u \in U$. The complement of F, denoted by F^c, is defined by $\mu_{F^c}(u) =$

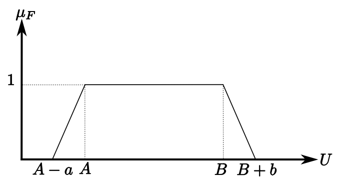


Fig. 1. Trapezoidal membership function

$1 - \mu_F(u)$. Furthermore, $F \cap G$ (resp. $F \cup G$) is defined the following way: $\mu_{F \cap G} = min(\mu_F(u), \mu_G(u))$ (resp. $\mu_{F \cup G} = max(\mu_F(u), \mu_G(u))$).

As usual, the logical counterparts of the theoretical set operators $\cap$, $\cup$ and complementation operator correspond respectively to the conjunction $\wedge$, disjunction $\vee$ and negation $\neg$. See [5] for more details.

2.2 About SQLf

The language called SQLf described in [2] extends SQL so as to support fuzzy queries. The general principle consists in introducing gradual predicates wherever it makes sense. The three clauses *select*, *from* and *where* of the base block of SQL are kept in SQLf and the "from" clause remains unchanged. The principal differences affect mainly two aspects :

– the calibration of the result since it is made with discriminated elements, which can be achieved through a number of desired answers (k), a minimal level of satisfaction (t), or both, and
– the nature of the authorized conditions as mentioned previously.

Therefore, the base block is expressed as:

select [**distinct**] [k | t | k, t] attributes
from relations
where fuzzy-cond

where "fuzzy-cond" may involve both Boolean and fuzzy predicates. This expression is interpreted as:

– the fuzzy selection of the Cartesian product of the relations appearing in the "from" clause,
– a projection over the attributes of the "select" clause (duplicates are kept by default, and if "distinct" is specified the maximal degree is attached to the representative in the result),
– the calibration of the result (top k elements and/or those whose score is over the threshold t).

The operations from the relational algebra — on which SQLf rests — are straightforwardly extended to fuzzy relations by considering fuzzy relations as fuzzy sets on the one hand and by introducing gradual predicates in the appropriate operations (selections and joins especially) on the other hand. The definitions of these extended relational operators can be found in [6]. As an illustration, we give the definitions of the fuzzy selection and join operators hereafter, where r and s denote two fuzzy relations defined respectively on the sets of domains X and Y.

– $\mu_{select(r,\ cond)}(t) = \top(\mu_r(t),\ \mu_{cond}(t))$ where *cond* is a fuzzy predicate and $\top$ is a triangular norm (most usually, *min* is used),
– $\mu_{join(r,\ s,\ A,\ B,\ \theta)}(tu) = \top(\mu_r(t),\ \mu_s(u),\ \mu_\theta(t.A,\ u.B))$ where A (resp. B) is a subset of X (resp. Y), A and B are defined over the same domains, θ is a binary relational operator (possibly fuzzy), $t.A$ (resp. $u.B$) stands for the value of t over A (resp. u over B).

3 An Extended Group-By Clause

In SQLf as in SQL, a *group by* clause builds a partition based on the (atomic) values of the attributes specified in this clause. Then, "*group by A*" leads to a partition where every group is associated with an A-value present in the relation. The idea we advocate here is to extend this mechanism so as to build a partition based on intervals or fuzzy sets of values.

3.1 Use of a Crisp Partition

The generic form of such a query is:

select label(A) [, aggregate, ...] **from** r [**where** ψ]
group by label(A)
using part(A) = $\{L_1, \ldots, L_n\}$

where $part(A)$ is a partition defined on the domain of A, $label(A)$ denotes any label L_i from $part(A)$, and ψ is a (fuzzy or crisp) condition.

Example 1. Let Emp be a relation of schema (id, nom, age, salary). Let us assume that one wants to retrieve the average salary for each age class (twenties, thirties, etc). In SQL, one has to express as many queries as there are age classes. However, one can imagine an expression of the type:

select label(age), **avg**(salary) **from** Emp
group by label(age)
using part(age) = $\{[20, 29], [30, 39], [40, 49], [50, 59]\}$.

With the data from Table 1, the result is:

$$\{\langle[20, 29], 2650\rangle, \langle[30, 39], 3200\rangle, \langle[40, 49], 4500\rangle, \langle[50, 59], 6100\rangle, \langle[60, 69], 3700\rangle\}. \quad \diamond$$

In the case where ψ is a fuzzy condition, the only type of aggregate which can appear in the *select* clause is *count*, because of the difficulty of defining the other aggregates on fuzzy sets. Indeed, the existing approaches dealing with the interpretation of aggregates in the general case [7] cannot be used in the framework of SQLf since they deliver a *fuzzy set* of possible evaluations (then, the lower and upper bounds of this fuzzy set of numbers can be computed). In SQLf, a unique degree of satisfaction attached to a condition such as $agg(A)$ is C — where agg denotes an aggregate and C a fuzzy condition — is needed in order to maintain compositionality.

For a given L_i from $part(A)$, $count(L_i)$ is computed as follows:

$$count(L_i) = \sum_{t \in r \,\wedge\, t.A \,\in\, L_i} \mu_\psi(t).$$

One also introduces a variant of *count*, denoted *count-rel*, which computes the average membership degree inside a group. It is defined as:

$$count\text{-}rel(L_i) = \frac{\sum_{t \in r \,\wedge\, t.A \,\in\, L_i} \mu_\psi(t)}{|\{t \in r \mid t.A \in L_i\}|}.$$

Table 1. Extension of relation emp

#e	e-name	position	age	w-dep	sal(k$)
17	Smith	engineer	51	3	65
76	Martin	engineer	40	5	45
26	Jones	secretary	24	3	19
12	Green	technician	39	3	32
19	Duncan	clerk	28	1	24
8	Brown	manager	54	1	57
31	Harris	technician	29	5	18
9	Davis	janitor	61	1	15
44	Howard	manager	22	3	45
23	Lewis	engineer	62	1	59

Example 2. Let us consider the query:

select label(age), **count**, **count-rel from** Emp
where salary **is** medium **group by** label(age)
using part(age) = $\{[20, 29], [30, 39], [40, 49], [50, 59]\}$.

With the fuzzy term *medium* defined as in Fig. 2 and the data from Table 1, one gets:

$$\{\langle[20, 29], 1, 0.25\rangle, \langle[30, 39], 0.7, 0.7\rangle, \langle[40, 49], 1, 1\rangle,$$
$$\langle[50, 59], 0.8, 0.4\rangle, \langle[60, 69], 0.6, 0.3\rangle\}. \qquad \diamond$$

3.2 Use of a Fuzzy Partition

The extension to the fuzzy partition case — which allows to take into account vague classes and to have a better robustness by making query results less sensitive to the boundaries of the classes — is rather straightforward. One just has to have available fuzzy partitions defined on the attribute domains. On the other hand, it is not possible anymore to use any type of aggregate in the *select* clause: one is limited to using *count* even when the condition in the *where* clause is Boolean (since the groups themselves are fuzzy). In the following, we assume that Ruspini partitions [8] are used, i.e.,

$$\forall x \in X, \sum_{L_i \in P} \mu_{L_i}(x) = 1$$

where P denotes a partition defined on domain X (cf. Fig. 2).

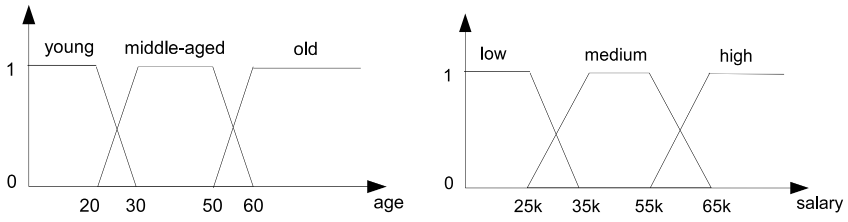


Fig. 2. Fuzzy partitions of the domains of attribute age (left) and salary (right)

Table 2. Tuples from emp and their degrees

#e	age	μ_{yg}	μ_{ma}	μ_{old}	sal	μ_{low}	μ_{med}	μ_{high}
17	51	0	0.9	0.1	65	0	0	1
76	40	0	1	0	45	0	1	0
26	24	0.6	0.4	0	19	1	0	0
12	39	0	1	0	32	0.3	0.7	0
19	28	0.2	0.8	0	24	1	0	0
8	54	0	0.6	0.4	57	0	0.8	0.2
31	29	0.1	0.9	0	18	1	0	0
9	61	0	0	1	15	1	0	0
44	22	0.8	0.2	0	45	0	1	0
23	62	0	0	1	59	0	0.6	0.4

Example 3. Let us consider the partition from Figure 2 (left) denoted by $part(age)$, and the query "find for each fuzzy age class the number of employees who earn more than \$30,000." It can be expressed as:

select label(age), **count from** Emp
where salary $>$ 30k **group by** label(age)
using part(age) = $\{$young, middle-aged, old$\}$.

With the data from Table 2, the result is: $\{\langle young, 0.8\rangle, \langle middle\text{-}aged, 3.7\rangle, \langle old, 1.5\rangle\}$ where each number returned corresponds to a Σ-count. $\diamond$

We now have the following definitions:

$$count(L_i) = \sum_{t \in r} \top(\mu_\psi(t),\ \mu_{L_i}(t.A)) \tag{1}$$

$$count\text{-}rel(L_i) = \frac{\sum_{t\,\in\,r} \top(\mu_\psi(t),\ \mu_{L_i}(t.A))}{\sum_{t\,\in\,r} \mu_{L_i}(t.A)}. \tag{2}$$

The fuzzy *group by* clause makes it possible to compute fuzzy summaries "on demand", in contrast with the approach proposed in [9] which first builds a summary of the whole database, then uses it to answer queries. An example of a query aimed at providing a fuzzy summary is given hereafter.

Example 4. Let us consider the query: "how are the ages of the employees whose salary is medium?". It can be expressed as:

select label(age), **count from** Emp
where salary **is** medium **group by** label(age)
using part(age) = $\{$young, middle-aged, old$\}$.

With the data from Table 2, and using $\top$ = *min*, one gets:

$$\{\langle young,\ 0.8\rangle,\ \langle middle\text{-}aged,\ 2.5\,\rangle, \langle old,\ 1.0\rangle\}. \qquad \diamond$$

Example 5. Let us now consider the query: "which proportion of employees from each age class has a salary which is medium?". It can be expressed as:

select label(age), **count-rel from** Emp
where salary **is** medium **group by** label(age)
using part(age) = {young, middle-aged, old}.

With the data from Table 2, and using ⊤ = *min*, one gets:

$$\{\langle young,\ 0.47\rangle,\ \langle middle\text{-}aged,\ 0.43\,\rangle,\ \langle old,\ 0.4\rangle\}. \qquad \diamond$$

4 Having Clause

The different forms of a *having* clause that can come as a complement to a *group-by* clause are described hereafter through a few examples.

4.1 Inclusion Constraint

An example which involves a Boolean condition in the *having* clause is: "find every age class such that at least 30% of the employees from that class have a high salary":

select label(age) **from** Emp E1
group by label(age)
having count ≥
  (**select count** * 0.3 **from** Emp **where** salary **is** high **and** age **is** E1.label(age))
using part(age) = {young, middle-aged, old}.

Another example, which involves a fuzzy *having* clause is: "find the extent to which all the employees of a given age class have a high salary:

select label(age) **from** Emp E1
group by label(age)
having (**select** #e **from** Emp **where** salary **is** high) **contains set**(#e)
using part(age) = {young, middle-aged, old}.

The evaluation of such a query rests on a graded inclusion. It constitutes the prototype expression for fuzzy association rule mining and will be detailed in Section 5.

4.2 Aggregate$_1$ θ Aggregate$_2$

Even though a limitation exists as to the aggregates which can be computed on a fuzzy set — as already mentioned —, it is still possible to evaluate conditions which *compare* two aggregates [10]. The idea is to start with a definition valid for crisp sets, then to extend it to fuzzy sets. In the case where A and B are crisp, it is possible to express the meaning of the statement $agg_1(A) \leq agg_2(B)$ using an implication according to the formula:

$$\forall x,\ [agg_1(A) \geq x] \Rightarrow [agg_2(B) \geq x] \tag{3}$$

where x is used to scan the definition domain of agg_1 and agg_2.

When A and B are two fuzzy sets, the expression $agg_1(A) \leq agg_2(B)$ (resp. $agg_1(A) \geq agg_2(B)$) is more or less satisfied. Its degree of truth $t(agg_1(A) \geq x)$

(resp. $t(agg_2(A) \geq x)$) can be obtained by considering the different α-level cuts of predicate A:

$$t(agg(A) \geq x) = max_{\alpha \in [0,\, 1]}\, min(\alpha,\ \mu_{\geq x}(agg(A_\alpha))).$$

Since "$\geq x$" is a Boolean predicate, its truth value is either 0 or 1 and we get:

$$t(agg(A) \geq x) = max_{\alpha \in [0,\, 1] such\ that\ agg(A_\alpha) \geq x}\ \alpha.$$

If the universal quantifier in (3) is interpreted as a generalized conjunction, the satisfaction degree of $agg_1(A) \leq agg_2(B)$ is given by:

$$min_{x \in D}\, t(agg_1(A) \geq x) \rightarrow t(agg_2(B) \geq x) \tag{4}$$

where $\rightarrow$ stand for a fuzzy implication [11] and D is the definition domain of agg_1 and agg_2. More detail can be found in [10].

An example of such a query is: "find the age classes such that the maximum of the technician's salaries is greater than the minimum of the engineer's salaries". It can be expressed as:

select label(age) **from** Emp E1 **where** job = 'technician'
group by label(age)
having max(salary) >
  (**select** min(salary) **from** Emp **where** age is E1.label(age) **and** job = 'engineer')
using part(age) = {young, middle-aged, old}.

In such a query, the *where* clause could also involve a fuzzy condition.

4.3 Aggregate is ψ

One may also compute the extent to which an aggregate satisfies a fuzzy condition ψ, by means of the approach proposed in [12]. For instance, when both the aggregate and the predicate ψ are increasing, one may start from the following definition of the statement $agg(A)\ is\ \psi$ when A is a crisp set and ψ is a Boolean condition:

$$agg(A)\ is\ \psi \Leftrightarrow \exists n \text{ such that } \psi(n) \text{ and } agg(A) \geq n. \tag{5}$$

When A is a fuzzy set and ψ is a fuzzy condition, the preceding formula may be generalized into:

$$t(agg(A)\ is\ \psi = max_{\alpha \in [0,\, 1]}\, min(\alpha,\ \mu_{\psi}(agg(A))). \tag{6}$$

The case where agg is not monotonous is more tricky, but an approach has also been proposed to deal with this situation, cf. [12].

An example of a query involving a condition of the form is $agg(A)\ is\ \psi$ inside a *having* clause is: "find the age classes where the average salary of engineers is high". It can be expressed as:

select label(age) **from** Emp **where** job = 'engineer'
group by label(age)
having avg(salary) **is** high
using part(age) = {young, middle-aged, old}.

Here again, the *where* clause may also involve a fuzzy condition.

5 Use for Association Rule Mining

We now explain how the fuzzy *group by* clause can be used for evaluating in a simple way fuzzy association rules of the type (*age* is $L_i \rightarrow$ *salary* is L'_j) where L_i and L'_j are two fuzzy labels defined respectively on the domain of *age* and that of *salary*. As mentioned in [13], at least two fuzzy extensions of association rules may be considered: those based on (fuzzy or scalar) cardinalities, and those based on fuzzy implications (cf. also [14] for the latter category). Hereafter, we do not deal with fuzzy-cardinality-based rules since they may be somewhat difficult to interpret for an end-user (and also because they cannot be represented and handled easily in a purely relational DBMS).

The approach based on a scalar cardinality (Σ-count) rests on a straightforward extension of the usual definition of confidence. Let us consider a fuzzy association rule of the type (A is $L_i \rightarrow B$ is L'_j). In this approach, the validity (confidence) of the rule is defined as:

$$\frac{\sum_{t \in r} \top(\mu_{L_i}(t.A),\ \mu_{L'_j}(t.B))}{\sum_{t \in r} \mu_{L_i}(t.A)}$$

In the fuzzy-implication-based approach, the rule (A is $L_i \rightarrow B$ is L'_j) expresses a constraint on the B-value for each tuple in the relation. The association rule, also denoted by $(A,\ L_i) \rightarrow (B,\ L'_j)$, then means: "for every tuple t, the more $t.A$ is L_i, the more $t.B$ is L'_j", i.e.:

$$(A,\ L_i) \rightarrow (B,\ L'_j) \Leftrightarrow \forall t \in r,\ \mu_{L_i}(t.A) \rightarrow_f \mu_{L'_j}(t.B)$$

where $\rightarrow_f$ denotes a fuzzy R-implication. In this case, the confidence of the rule is equal to:

$$min_{t \in r}\ \mu_{L_i}(t.A) \rightarrow_f \mu_{L'j}(t.B).$$

5.1 Rules of the Type A Is $L_i \rightarrow B$ Is L'

Computation of the Support. First, one introduces a variant of *count* named *count-g* whose general definition is:

$$count\text{-}g(L_i) = \frac{\sum_{t\,\in\, r} \top(\mu_\psi(t),\ \mu_{L_i}(t.A))}{|r|}. \tag{7}$$

Let us consider the generic SQLf query:

select label(A), **count-g from** r **where** B **is** L'
group by label(A)
using p(A) = $\{L_1,\ \ldots,\ L_n\}$

It allows for computing the support of every fuzzy association rule of the type "A is L_i $\rightarrow B$ is L'" for a given L'.

Computation of the Confidence. Let us consider the generic SQLf query:

select label(A) **from** r
group by label(A)
having (**select** K **from** r **where** B **is** L') **contains set**(K)
using p(A) = $\{L_1, \ldots, L_n\}$

where K denotes the primary key of the relation.

Scalar Cardinality Using the approach based on scalar cardinality, the confidence of the rule corresponds to the *cardinality-based* degree of inclusion of the fuzzy set:

$E(L_i)$ = (**select** #e **from** r **where** A **is** L_i)

in the fuzzy set:

F = (**select** #e **from** r **where** B **is** L')

i.e., to the degree μ produced by the evaluation of the *having* clause:

$$\mu = \frac{\sum_{t \in r} \top(\mu_{L'}(t.B),\ \mu_{L_i}(t.A))}{\sum_{t \in r} \mu_{L_i}(t.A)}.$$

R-implication. Using the approach based on a fuzzy R-implication, the confidence of the rule corresponds to the *implication-based* degree of inclusion of $E(L_i)$ in F — see, e.g., [15] —, i.e. to the degree μ produced by the evaluation of the *having* clause when *contains* is replaced by *contains-f*:

$$\mu = min_{t \in r}\ \mu_{L_i}(t.A) \rightarrow_f \mu_{L'}(t.B).$$

For instance, with Łukasiewicz' implication (*contains-Lu*), we have:

$$\mu = min_{t \in r}\ min(1,\ 1 - \mu_{L_i}(t.A) + \mu_{L'}(t.B)).$$

Example 6. Let us consider the set of rules of the form:

$$age \text{ is } L_i \rightarrow salary \text{ is } medium$$

where L_i belongs to $p(age)$. They can be evaluated by means of the following two queries:

select label(age), **count-g from** Emp **where** salary **is** medium
group by label(age)
using p(age) = {young, middle-aged, old};

which, using the data from Table 1, the partitions from Fig. 2, and $\top$ = *min* returns:

$$\{0.08/\langle \textit{young} \rangle, 0.25/\langle \textit{middle-aged} \rangle,\ 0.1/\langle \textit{old} \rangle\}.$$

and

select label(age) **from** Emp **group by** label(age)
having (**select** #e **from** Emp **where** salary **is** medium)
    contains set(#e)
using p(age) = {young, middle-aged, old}

which returns:

$$\{0.47/\langle \textit{young} \rangle, 0.43/\langle \textit{middle-aged} \rangle,\ 0.4/\langle \textit{old} \rangle\}.$$

If *contains* were replaced by *contains-Lu* in the second query, we would get:

$$\{0.4/\langle \textit{young} \rangle, 0.1/\langle \textit{middle-aged} \rangle, 0/\langle \textit{old} \rangle\}. \qquad \diamond$$

5.2 Rules of the Type A Is $L \rightarrow$ B Is L'_i

Now let us consider mining fuzzy association rules of the type "A is $L \rightarrow$ B is L'_i" for a given L. Again, two SQLf queries are necessary:

select label(B), **count-g from** r **where** A **is** L
group by label(B)
using p(B) = $\{L'_1, \ldots, L'_n\}$

for computing the support of the rules considered, and:

select label(B) **from** r **group by** label(B)
having set(K) **contains** (**select** K **from** r **where** A **is** L)
using p(B) = $\{L'_1, \ldots, L'_n\}$

for computing their confidence values.

Example 7. Let us consider the set of rules of the form:

$$age \text{ is } young \rightarrow salary \text{ is } L'_i$$

where L'_i belongs to $p(salary)$. They can be evaluated by means of the following two queries:

select label(salary), **count-g from** Emp **where** age **is** young **group by** label(salary)
using p(salary) = {low, medium, high}

and

select label(salary) **from** Emp **group by** label(salary)
having set(#e) **contains**
    (**select** #e **from** Emp **where** age **is** young)
using p(salary) = {low, medium, high}.

With the data from Table 1 and the partitions from Fig. 2, the first query returns:

$$\{0.09/\langle \textit{low} \rangle, 0.08/\langle \textit{medium} \rangle, 0/\langle \textit{high} \rangle\}.$$

As to the second one — which corresponds to the scalar cardinality approach since it involves the operator *contains* —, it returns the result:

$$\{0.53/\langle \textit{low} \rangle, 0.47/\langle \textit{medium} \rangle, 0/\langle \textit{high} \rangle\}.$$

If *contains* were replaced by *contains-Lu* in the second query, we would get:

$$\{0.2/\langle \textit{low} \rangle,\ 0.4/\langle \textit{medium} \rangle,\ 0.2/\langle \textit{high} \rangle\} \qquad \diamond$$

6 About the Evaluation of a Fuzzy Group-By

The complexity of the evaluation of an FGB clause is very similar to that of a regular group-by clause. Two cases may be distinguished:

- if the attribute appearing in the FGB clause — let us denote it by A — is indexed, it is possible to directly access the tuples which belong to a given fuzzy class L_i: they are the tuples t such that $t.A \in \mathit{support}(L_i)$. Let us recall that the support of a fuzzy label is expressed as an interval and can be straightforwardly determined from the membership function associated with that label. The degree to which tuple t belongs to class L_i is equal to $\mu_{L_i}(t.A)$.
- otherwise, as usual, one may sort the relation on attribute A and compare the A-value of each tuple with the (overlapping) segments which correspond to the supports of the different fuzzy labels in the partition of $\mathit{domain}(A)$ in order to build the fuzzy groups. Here too, of course, $\mu_{L_i}(t.A)$ must be computed for each tuple t.

As can be seen, the cost of the evaluation of an FGB clause should be more or less equivalent to that of a regular group-by clause since the only additional cost is that related to the computation of the membership degrees.

7 Related Work

7.1 Extended Group-By

The work in [16] proposes some SQL constructs to make clustering facilities available from SQL in the context of spatial data. Basically, these constructs act as wrappers of conventional clustering algorithms but no further integration with database systems is studied. Li *et al.* [17] extend the group-by operator to approximately cluster all the tuples in a predefined number of clusters. Their framework makes use of conventional clustering algorithms, e.g. K-means, and employ summaries and bitmap indexes to integrate clustering and ranking into database systems. Silva *et al.* [3] introduce a similarity group-by operator in order to group objects with similar values. Our study differs from [17] and [3] in that (1) we focus on fuzzy grouping based on vague concepts, not on similarity-based grouping; (2) we do not aim at "discovering" the clusters, since in our approach the groups are explicitly specified in the query (by means of a fuzzy partition), which by the way gives them a well-identified meaning; and (3) the authors of [17] and [3] do not consider a general fuzzy querying framework such as SQLf (where the fuzzy group-by construct is just a piece of the puzzle) but only extend a particular feature of SQL.

7.2 Fuzzy OLAP

A few research works, e.g. [18,19], have been devoted to the introduction of fuzziness into OLAP systems. These approaches share some common characteristics with ours (use of fuzzy partitions, fuzzy association rule mining) but they do not rely on a general purpose database querying language such as SQL(f). They rather extend operators such as *roll-up* and *drill-down*, or devise specific rule mining algorithms.

7.3 Fuzzy Database Summarization Techniques

Developed by Rasmussen and Yager, SummarySQL [20] is a fuzzy query language which can evaluate the truth degree of a summary guessed by the user. A summary expresses knowledge about the database in a statement under the form "Q objects in DB are S" or "Q R objects in DB are S" where DB stands for the database, Q is a linguistic quantifier and R and S are linguistic terms. The expression is evaluated for each tuple and the associated truth values are later used to obtain a truth value for the summary. The statements considered by the authors are in a sense more general than the fuzzy association rules that we deal with, since they involve fuzzy quantifiers. However, our approach can easily be extended to capture such statements by relaxing the operator *contains* that appear in the *having* clause, using e.g. one of the approaches described in [15]. When it comes to mining fuzzy statements, the main difference lies in the fact that [20] does not propose any SQL construct to evaluate these statements "in a batch" as we do thanks to the FGB clause: the statements have to be checked one by one and no fuzzy partitioning of the domains is used.

In [9], Saint-Paul *et al.* propose an approach to the production of linguistic summaries structured in a hierarchy, i.e., a summarization tree where the tuples from the database are rewritten using the linguistic variables involved in fuzzy partitions of the attribute domains. The main difference with our approach is that [9] views summarization as an independent process, which is not performed by means of SQL queries but by a specific algorithm. As mentioned before, the FGB operator enables to obtain summaries "on demand" without having to summarize the whole database.

7.4 Mining Association Rules with SQL

The use of SQL queries for mining association rules has been advocated by several authors, see e.g. [21,22,23,24,25,26,27]. However, none of these approaches considers an extended group-by mechanism, and none considers *fuzzy* association rules either. To the best of our knowledge, the only approach which uses a fuzzy extension of SQL for mining fuzzy association rules (or gradual functional dependencies, as the authors call them) is [28], which relies on SummarySQL already discussed in the subsection above.

8 Conclusion

In this paper, we have introduced a fuzzy group-by (FGB) operator based on the use of fuzzy partitions of attribute domains and described how it could be integrated into the SQLf language. The main goal of FGB is to generate more meaningful and useful groupings than the regular group-by operator. We have shown how this construct makes it possible to generate fuzzy summaries "on demand', as well as to mine fuzzy association rules in a practical way.

Among perspectives for future work, let us mention:

– implementation aspects and experimentations: it is of course important to make sure that queries involving a fuzzy grouping have execution times comparable to those involving a classical group-by. One can be reasonably optimistic about this issue given the results presented in [3] about a similarity-based group-by (SGB), which show that the overhead in this case is no more than 25%. FGB should be even more efficient than SGB since i) the clusters are predefined, ii) the use of fuzzy partitions still makes it possible to employ evaluation techniques based on sorts and/or indexes (cf. Section 6).
– an investigation about the way other measures than support and confidence for assessing fuzzy association rules, cf. [29], could be taken into account;
– an extension of the format of the rules to be mined, for instance through a relaxation of the universal quantifier based on one of the approaches proposed in [15].

References

1. Tahani, V.: A conceptual framework for fuzzy query processing — a step toward very intelligent database systems. Information Processing and Management 13(5), 289–303 (1977)
2. Bosc, P., Pivert, O.: SQLf: a relational database language for fuzzy querying. IEEE Transactions on Fuzzy Systems 3(1), 1–17 (1995)
3. Silva, Y.N., Aref, W.G., Ali, M.H.: Similarity group-by. In: Proc. of ICDE 2009, pp. 904–915 (2009)
4. Zadeh, L.A.: Fuzzy sets. Information and control 8(3), 338–353 (1965)
5. Dubois, D., Prade, H.: Fundamentals of fuzzy sets. The Handbooks of Fuzzy Sets, vol. 7. Kluwer Academic Pub., Netherlands (2000)
6. Bosc, P., Buckles, B., Petry, F., Pivert, O.: Fuzzy databases. In: Bezdek, J., Dubois, D., Prade, H. (eds.) Fuzzy Sets in Approximate Reasoning and Information Systems. The Handbook of Fuzzy Sets Series, pp. 403–468. Kluwer Academic Publishers, Dordrecht (1999)
7. Dubois, D., Prade, H.: Measuring properties of fuzzy sets: a general technique and its use in fuzzy query evaluation. Fuzzy Sets and Systems 38(2), 137–152 (1990)
8. Ruspini, E.H.: A new approach to clustering. Information and Control 15(1), 22–32 (1969)
9. Saint-Paul, R., Raschia, G., Mouaddib, N.: General purpose database summarization. In: Proc. of VLDB 2005, pp. 733–744 (2005)
10. Bosc, P., Pivert, O., Liétard, L.: On the comparison of aggregates over fuzzy sets. In: Bouchon-Meunier, B., Foulloy, L., Yager, R. (eds.) Intelligent Systems for Information Processing: From Representation to Applications, pp. 141–152. Elsevier, Amsterdam (2003)
11. Fodor, J., Yager, R.: Fuzzy-set theoretic operators and quantifiers. In: Dubois, D., Prade, H. (eds.) Fundamentals of Fuzzy Sets. The Handbooks of Fuzzy Sets Series, vol. 1, pp. 125–193. Kluwer Academic Publishers, Dordrecht (2000)
12. Bosc, P., Liétard, L.: Aggregates computed over fuzzy sets and their integration into SQL. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 16(6), 761–792 (2008)

13. Bosc, P., Pivert, O.: On some fuzzy extensions of association rules. In: Proc. of the Joint 9th IFSA World Congress and 20th NAFIPS International Conference, Vancouver, Canada, pp. 1104–1109 (2001)
14. Hüllermeier, E.: Implication-based fuzzy association rules. In: Siebes, A., De Raedt, L. (eds.) PKDD 2001. LNCS (LNAI), vol. 2168, pp. 241–252. Springer, Heidelberg (2001)
15. Bosc, P., Pivert, O.: On two qualitative approaches to tolerant inclusion operators. Fuzzy Sets and Systems 159(21), 2786–2805 (2008)
16. Zhang, C., Huang, Y.: Cluster by: a new SQL extension for spatial data aggregation. In: Proc. of ACM GIS, pp. 53–56 (2007)
17. Li, C., Wang, M., Lim, L., Wang, H., Chang, K.C.C.: Supporting ranking and clustering as generalized order-by and group-by. In: Proc. of SIGMOD 2007, pp. 127–138 (2007)
18. Delgado, M., Molina, C., Ariza, L.R., Sánchez, D., Miranda, M.A.V.: F-cube factory: a fuzzy olap system for supporting imprecision. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 15(Suppl. 1), 59–81 (2007)
19. Kaya, M., Alhajj, R.: Online mining of fuzzy multidimensional weighted association rules. Appl. Intell. 29(1), 13–34 (2008)
20. Rasmussen, D., Yager, R.R.: Summary SQL – a fuzzy tool for data mining. Intell. Data Anal. 1(1-4), 49–58 (1997)
21. Meo, R., Psaila, G., Ceri, S.: An extension to SQL for mining association rules. Data Min. Knowl. Discov. 2(2), 195–224 (1998)
22. Clear, J., Dunn, D., Harvey, B., Heytens, M.L., Lohman, P., Mehta, A., Melton, M., Rohrberg, L., Savasere, A., Wehrmeister, R.M., Xu, M.: Nonstop SQL/MX primitives for knowledge discovery. In: Proc. of KDD 1999, pp. 425–429 (1999)
23. Thomas, S., Sarawagi, S.: Mining generalized association rules and sequential patterns using SQL queries. In: Proc. of KDD 1998, pp. 344–348 (1998)
24. Yoshizawa, T., Pramudiono, I., Kitsuregawa, M.: SQL based association rule mining using commercial RDBMS (IBM DB2 UDB EEE). In: Kambayashi, Y., Mohania, M., Tjoa, A.M. (eds.) DaWaK 2000. LNCS, vol. 1874, pp. 301–306. Springer, Heidelberg (2000)
25. Imielinski, T., Virmani, A.: MSQL: A query language for database mining. Data Min. Knowl. Discov. 3(4), 373–408 (1999)
26. Rajamani, K., Cox, A.L., Iyer, B.R., Chadha, A.: Efficient mining for association rules with relational database systems. In: Proc. of IDEAS 1999, pp. 148–155 (1999)
27. Pereira, R., Millan, M., Machuca, F.: New algebraic operators and SQL primitives for mining association rules. In: Neural Networks and Computational Intelligence, pp. 227–232 (2003)
28. Rasmussen, D., Yager, R.R.: Finding fuzzy and gradual functional dependencies with SummarySQL. Fuzzy Sets and Systems 106(2), 131–142 (1999)
29. Dubois, D., Hüllermeier, E., Prade, H.: A systematic approach to the assessment of fuzzy association rules. Data Min. Knowl. Discov. 13(2), 167–192 (2006)

OLAP Operators for Complex Object Data Cubes

Doulkifli Boukraâ[1], Omar Boussaïd[2], and Fadila Bentayeb[2]

[1] High School of Computer Science, Oued-Smar, Algiers
d_boukraa@esi.dz

[2] Lumière University - Lyon 2, 5 avenue Pierre Mendès-France, 69676 Bron Cedex
{omar.boussaid,fadila.bentayeb}@univ-lyon2.fr

Abstract. Nowadays, multidimensional models are recognized to best reflect the decision makers' analytical view of data. The classical multidimensional models were meant to analyze conventional data (numerical and categorical). However, they fail to handle data complexity, which is expressed by the multiplicity of data sources, the heterogeneity of formats, the diversity of structures, etc. To this end, new multidimensional models have been proposed for OLAP purposes. Nevertheless, data complexity is partially covered in these models, which may cause a lack in decision making. In our previous work, we proposed to integrate data complexity within a complex object-based multidimensional model. In this paper, based on our proposed model, we provide adapted OLAP operators that take into account data complexity. Thus, we define operators to create complex data cubes, to visualize them and to analyze them.

Keywords: Multidimensional model, complex object, complex cube, OLAP operator.

1 Introduction

1.1 Context and Related Work

Nowadays, multidimensional modeling is recognized to best reflect the decision makers' analytical view of data as witnessed by the literature richness about multidimensional models. These models were surveyed in [1]. Associated with the models are the OLAP operators that allow expressing analysis needs such as *slice-and-dice*, *rollup* and *drill-down* [5]. Besides, decision making involves more and more complex data (multiple sources, heterogeneous formats, diverse structures, etc.) Warehousing and analyzing complex data are not straightforward activities. Moreover, we believe that the more data complexity aspects are considered in the warehousing process, the more accurate decisions are.

Recently, there have been several papers on warehousing and analyzing non-conventional data. Examples of related work deal with unstructured textual data [9], semistructured data, represented with XML [8], temporal data [16], spatial data [7].

Regarding data modeling and analysis, three approaches can be distinguished. The first approach consists in using existing OLAP tools to analyze non conventional data. In this case, data may remain in their sources as in [10] and OLAP operates on a integrated virtual schema of a middleware. User queries are then translated according to the data source schemes. Some other papers propose to capture the multidimensional concepts from non-conventional data in a bottom-up way and provide multidimensional models that can integrate into existing OLAP tools [8]. The advantage of this approach is to benefit from the maturity of existing OLAP technologies. However, data complexity is lost due to the limitation of the underlying multidimensional models. A mixed solution consists in extending traditional OLAP querying to external object data [14]. In this case, the object data serves as a decoration of the retrieved multidimensional data.

In the second approach, new multidimensional models are provided to deal with one or many aspects of data complexity. Some models are brought to the conceptual level such as object-models [12,13], temporal data models [16] or spatial models [2]. Other models are described at the logical level, especially with XML for semistructured data. Besides, the underlying OLAP operations are revisited with respect to data nature and new operators are proposed. Examples include the XML OLAP operators [17] and textual data aggregation [15].

1.2 Motivation and Contributions

The related work shows that many aspects of data complexity are covered, yet separately. Furthermore, there is a lack of a framework that integrates as many aspects as possible. We believe that such a framework would leverage the decision making process since it provides the analysts with different points of view of the same data, which is likely to be the case in real life. For instance, to best diagnose a medical case, doctors would combine numerical data (e.g. measurements) with textual reports, radiographies, etc. Moreover, a complex data-warehousing and analysis framework has to address the following issues:

– Cover as many data complexity aspects as possible in the multidimensional model;
– Integrate into existing OLAP tools when only some aspects of complexity are considered;
– Support a large set of OLAP visualization techniques.

In a previous work, we proposed a multidimensional model that addresses the first issue [3]. Our model is based on the concept of *complex object* that covers many aspects of data complexity, such as the multiplicity of structures, formats, sources, etc. In this paper, our main contributions are the following:

– a set of OLAP operators to construct complex data cubes;
– a set of OLAP operators to visualize the data cubes and to analyze them.

The remainder of this paper is organized as follows. In section 2, we recall the concepts of our proposed model. Then, we present in section 3 the set of operators that construct complex cubes, visualize their data and analyze them. In section 4, we present some implementation details. Finally, we conclude in section 5 and give some perspectives.

2 The Complex Object-Based Multidimensional Model

In this section, we recall the main concepts underlying our multidimensional model of complex objects. Further details can be found in [3].

2.1 Concepts and Definitions

The Complex Object. The concept of complex object (CO) was proposed by Boussaïd et al [4] as a solution for complex data integration. According to the authors, a CO is a physical or abstract entity composed by one or many sub-documents[1]. Each sub-document may represent a simple or tagged text, a relational view, an image or temporal data (e.g. sound, video). Basically, a CO describes low level features of data (e.g. image color) and other features such as the object's source name and languages, but it can be extended to include other features such as semantic information (e.g. image content). In our model, we use a CO to represent both facts and dimension members. Compared to existing object multidimensional models, a fact or dimension member can be represented with a whole UML class diagram rather than a single UML class as in many models. The advantage of such a conceptualization is the possibility to describe structurally and semantically rich facts and dimensions. A CO fact is equivalent to the xFact introduced by Nassis et al. [13]. In addition, we abstract a CO as a set of attributes. An attribute may be simple (e.g. image color) or complex if composed by simple or other complex attributes (e.g. a whole image). Attributes are then related to each other via several relationships such as composition and association. At this stage, however, we consider only one kind of relationships that organizes attributes in hierarchies, as it will be seen later.

Definition 1. A CO is a pair $Obj = (ID^{Obj}, SA^{Obj})$ where ID^{Obj} represents the object's identifier and $SA^{Obj} = \{A_i^{Obj} / i \in \mathbb{N}\}$ represents the set of its attributes.

The Complex Relationship. A complex relationship (CR) is an explicit link between two COs. It may range from simple associations to aggregations, compositions and specialization/generalization, etc. A CR is characterized by its name and by the names of the two COs that it links.

Definition 2. A CR is a pair $R = (Obj_s^R, Obj_t^R)$ where Obj_s^R represents the source object of R and Obj_t^R represents its target object.

[1] The term *document* is used in a broad sense.

The Attribute Hierarchy. An attribute hierarchy (AH) is a special relationship between a CO's attributes. It is characterized by its name and by the set of its attributes, each one having a level within the hierarchy.

Definition 3. An AH is denoted by $AH^{Obj} = \{A_i^{Obj} \in SA^{Obj} \cup \{ID^{Obj}\}/i \in \mathbb{N}\} \cup \{All^A\}$ where All^A denotes a dummy attribute at the least detailed level.

The Object Hierarchy. An object hierarchy (OH) is similar to an AH but it is defined between many COs rather than between attributes. An OH is characterized by its name and by the set of COs, each one having a level within the hierarchy.

Definition 4. An OH is denoted by $OH = \{Obj_i/i \in \mathbb{N}\} \cup \{All^{Obj}\}$ where All^{Obj} represents a dummy object at the least detailed level.

The Multidimensional Schema. The multidimensional schema is composed by: (1) the set of complex objects, (2) the set of complex relationships, (3) the set of attribute hierarchies and (4) the set of object hierarchies.

Definition 5. The multidimensional schema is denoted by SCM=(SO, SR, SAH, SOH) where $SO = \{Obj_i/i \in \mathbb{N}\}, SR = \{R_j/j \in \mathbb{N}\}, SAH = \{AH_k/k \in \mathbb{N}\}$ and $SOH = \{OH_m/m \in \mathbb{N}\}$.

2.2 Example 1

A research laboratory wishes to warehouse data about scientific publishing in order to answer different analysis needs like (1) assessing the quality of publications according to different criteria (e.g. publication ratings), (2) assessing the scientific production of a researcher according to his/her publishing frequency. Data of scientific publishing can be considered as complex: they originate from many sources (e.g. DBLP, PubZone.org), they may have different formats (e.g. images in conference websites) and they may be diversely structured (e.g. publications are typically semistructured). In order to meet the users' analysis needs, the data may be organized using our model as follows.

1. The complex objects: *Publication*, *Author*, *Proceedings*, *Conference*, *Jounal_number*, *Journal_volume*, *Journal*, *Date*. Examples of attributes for the object *Publication* are *title*, *pages*, *keyword*, *type* and the identifier *publication_id*;
2. The relationships: *Authored_by* between *Publication* and *Author*, *Date_pub* between *Publication* and *Date*, *Publi_conf* between *Publication* and *Proceedings*, *Publi_journal* between *Publication* and *Journal_number*;
3. The attribute hierarchies: *H_pub* associated with *Publication* and composed by *publication_id* and *type*, *H_time* associated with *Date* and composed by *date_id*, *month* and *year*;

4. The object hierarchies : *H_conf* composed by *Proceedings* and *Conference*, *H_journal* composed by *Journal_number*, *Journal_volume* and *Journal*.

The multidimensional schema described above is depicted in Figure 1.

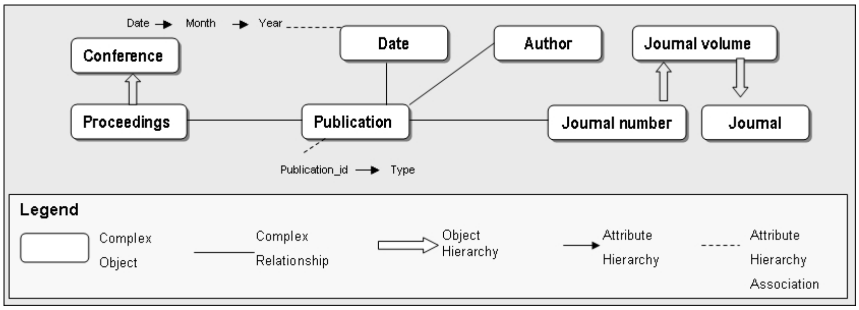


Fig. 1. Example of a complex object-based multidimensional schema

3 OLAP Operators

Our proposed multidimensional model is independent from any analysis context, i.e. there is no a priori fact or dimension. Indeed, the fact and dimensions are defined on-line using a projection operation of the multidimensional schema on a set of its components. The projection produces a new structure, called *complex cube* that can be materialized and visualized. Furthermore, existing cubes can serve as the basis to create new ones, by modifying either their structure or their data. In this section, we present all these operations.

3.1 The Cube Construction Operators

Cubic Projection. The objective of this operation is to construct a complex cube from the multidimensional schema by projecting it on the following elements:

1. one complex object to play the role of the fact;
2. a set of measures, each one associated with
   (a) one attribute of the fact containing the basic (most detailed) values of the measure;
   (b) one function that aggregates the measure values;
   (c) a set of relationships along which the aggregation of the measure values makes sense (i.e. respecting the measure additivity);
3. a set of relationships that link the fact to the other objects;
4. a set of objects that are directly linked to the fact;
5. a set of object hierarchies containing the projected objects in 4;
6. a set of object hierarchies associated with the objects projected in 4 and 5.

A dimension is then composed by the members of all the object hierarchies that contain the object directly linked to the fact.

Definition 6. Let $SCM = (SO, SR, SAH, SOH)$ be a multidimensional schema and $SAF = \{af_i, i \in N\}$ be a set of aggregation functions. The cubic projection is denoted by $\Pi_C Obj(SCM) = C = (F, SM, SR^C, SD, SAH^C, SOH^C)$ where

- F is the fact object such that $F \in SO$;
- SM is the set of measures such that $SM = \{M_i / i \in \mathbb{N}\}$ where M_i represents a measure. We also define the following three functions. (1) $AttM$ associates each measure M_i with one of the fact attributes, denoted by A^{M_i} where $A^{M_i} \in \{ID^F\} \cup SAF$. (2) AggRel associates each measure M_i with the set SR^{M_i}. The set SR^{M_i} is such that $SR^{M_i} = SR^C$ if M_i is additive, $SR^{M_i} = \phi$ if M_i is non-additive and $SR^{M_i} \subset SR^C$ if Mi is semi-additive. (3) AggFun associates each measure M_i with a function $af^{M_i} \in SAF$;
- $SR^C = \{R_i^C, i \in \mathbb{N}\}$ is the set of relationships where $SR^C \subseteq SR$;
- $SD = \{D_j, j \in \mathbb{N}\}$ is the set of dimensional objects $SD \subseteq SO$;
- $SAH^C = \{OH_m^C, m \in \mathbb{N}\}$ is the set of reduced attribute hierarchies;
- $SOH^C = \{AH_k^C, k \in \mathbb{N}\}$ is the set of reduced object hierarchies.

Example 2. Consider the multidimensional schema (Fig. 1) called *SCM_pub*. Let's suppose that the user aims at analyzing the publication ratings and their keywords to get the maximum ratings of publications by author and period of time and the top keywords by author and conference. The corresponding cube for this analysis context is the result of projecting *SCM_Pub* on the CO *Publication*. We denote by *C_pub* such a cube. Then, *C_pub*= Π_C *Publication*(*SCM_pub*)= (*F*, *SM*, $SR^{C\text{-}pub}$, *SD*, $SAH^{C\text{-}pub}$, $SOH^{C\text{-}pub}$) such that

- *F* = *Publication*
- *SM* = {*max_rating*, *top_keyword*} where
  - AttM(*max_rating*) = *Rating*
  - AggRel(*max_rating*) = {*Authored_by*, *Date_pub*}
  - AggFun(*max_rating*) = max
  - AttM(*top_keyword*) = *keyword*
  - AggRel(*top_keyword*) = {*Authored_by*, *Publi_conf*}
  - AggFun(*top_keyword*) = *top_keyword*
- $SR^{C\text{-}pub}$={*Authored_by*, *Date_pub*, *Publi_journal*, *Publi_conf*}
- *SD* = {*Time*, *Author*, *Proceeding*, *Conference*, *Journal_number*, *Journal_volume*, *Journal*}
- $SAH^{C\text{-}pub}$ = {*H_time*}
- $SOH^{C\text{-}pub}$ = {*H_conf*, *H_journal*}

Constructing new cubes from existing cubes. In our model, a complex cube is a materialized view of the multidimensional schema. Materialized views (MV) are used to significantly enhance query response time in data warehouses [11]. In our work, we use MV (existing cubes) to construct new cubes as an

Table 1. Structure-related operators for complex cube construction

Operation	**Definition**
Add / remove a relationship	$ADD_R(C, R_+) \| REM_R(C, R_-^C)$
Add / remove an attribute hierarchy	$ADD_{AH}(C, AH_+) \| REM_{AH}(C, AH_-^C)$
Add / remove an object hierarchy	$ADD_{OH}(C, OH_+) \| REM_{OH}(C, OH_-^C)$
Add / remove a measure	$ADD_M(C, M_+) \| REM_M(C, M_-^C)$

Table 2. Data-related operators for complex cube construction

Operation	**Definition**
Data selection according to a predicate on an object	$\sigma_C C(P(Obj_\sigma))$
Union of two cubes	$C_1 \cup C_2$
Difference of two cubes based one object	$C_1 -_C C_2(Obj_-)$
Intersection of two cubes based on one object	$C_1 \cap C_2(Obj_\cap)$

alternative to using the multidimensional schema. This is argued by the following. First, from a structural standpoint, the analysis contexts are likely to share many common elements (the same fact, the same dimensions, etc.) and it is natural to modify the structure of existing cubes by adding or deleting some elements. Secondly, from a content standpoint, many analysis contexts may correspond to the same cube structure while containing different data. In this section, we provide two kinds of cube-based operators: (1) structure-related operations (table 1) that modify the structure of existing cubes and (2) data-related operations (table 2) that produce same-structured cubes but that contain different data.

Example 3. Based on the cube *C_pub* of example 2, the user can create a new cube in order to analyze *max_rating* by *author* and by *date*. We can write C_pub_1 = REM_{OH} (REM_{OH} (REM_R (REM_R (REM_M(*C_pub*, *top_keyword*), *Publi_journal*),*Publi_conf*),*H_conf*),*H_journal*). Now, based on C_pub_1, the user can switch from *max_rating* to *top_keyword* and analyze it by *author* and by *journal*. We can write C_pub_2 = ADD_{OH} (ADD_R (REM_R (REM_M (ADD_M (C_pub_1, *top_keyword*), *max_rating*), *Date_pub*), *Publi_journal*), *H_journal*). Then, based on *C_pub*2, the user can create two cubes that contain respectively publications whose titles contain the word *database*: $C_1 = \sigma_C(Contains(Publication.Title, =' database'))$ and authors whom surnames begin with the letter A: $C_2 = \sigma_C(FirstLetter(Author.FamilyName =' A')))$. Finally, based on C_1 and C_2, a cube can be created for publications whose titles contain the word *database* and written, among others, by authors whom surnames begin with A:$C_1 \cap C_2(Publication)$.

3.2 Visualization Operators

In order to perform OLAP analyzes, data are displayed using different visualization solutions (e.g. cross-tabs). Choosing a convenient solution for the user is an issue in OLAP visualization [6]. Therefore, as stated in [6], there is a need of an abstraction layer of OLAP visualization that enables switching from one visualization technique to another. In this paper, we introduce the notion of view over a complex cube as an abstract visualization solution. Thus, it is possible to describe OLAP analysis operations at an abstract level and then let the user choose the appropriate interface according the nature of data to be analyzed. Furthermore, we provide a formal description of a view that can be stored and grouped with other view definitions to form navigation contexts. Then, the navigation contexts can be further processed using different techniques (e.g. data mining) to enhance or personalize the OLAP user interface.

View Projection. A view projection operation is similar to the *Display* operation introduced in [15]. In our work, this operation displays the following elements:

- A view fact (VF) that maps onto to the fact of the complex cube. A VF is characterized by its name and by the set of its features. A feature maps onto a fact attribute of the complex cube;
- A set of measures, selected among the measures of the cube;
- A view dimension (VD) per relationship of the cube. A VD corresponds to a dimension of the cube and it is characterized by its name and by the set of its features. A VD feature maps onto one attribute of a dimension member of the cube. Moreover, in order to know the aggregation level of the measure values, we associate the VD with two elements:
    - a complex object that belongs to one object hierarchy of the dimension if there is any. We denote by AO such an object. In case there is no hierarchy, the VD is associated with the object directly linked to the fact;
    - an attribute that belongs to one attribute hierarchy related to AO. Let us call this attribute AA.

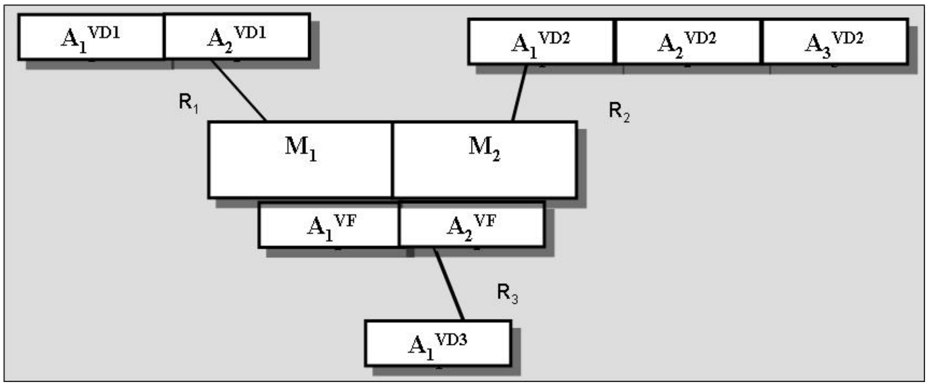


Fig. 2. The notion of view over a complex cube

The aggregated values of a measure are calculated along the AO then along AA. The VD features that may be displayed on a view depend on AO and AA. Figure 2 depicts the notion of view over a complex cube. In this figure, the VF features are A_1^{VF} and A_2^{VF}, the measures are M_1 and M_2, the VDs are VD_1, VD_2 and VD_3 which correspond to the relationships R_1, R_2 et R_3 and the VD features are $A_1^{VD_1}$, $A_2^{VD_1}$, $A_1^{VD_2}$, $A_2^{VD_2}$, $A_3^{VD_2}$ and $A_1^{VD_3}$.

Definition 7. Let $C = (F, SM, SR^C, SD, SAH^C, SOH^C)$ be a complex cube. The view projection is denoted by $V(C) = V^C = (F^V, SM^V, SD^V)$ where

- F^V is the VF such that $F^V = \{A_p^{FV} / p \in \mathbb{N}\}$ where $F^V \subseteq SA^F \cup ID^F$ if the view displays the basic data of the cube (no aggregation is performed) and it is set to a constant *Undefined* if the view displays aggregated values of at least one measure;
- $SM^V = \{M_i^V / i \in \mathbb{N}\} \subseteq SM$ is the set of measures to be displayed;
- $SD^V = \{D^{VR_j} / j \in \mathbb{N}\}$ where D^{VR_j} is a view dimension corresponding to the relationship R_j of C. Moreover, we define the function $ViewObj(D^{VR_j})$ $= VO^{VR_j}$ which associates D^{VR_j} with an object belonging to the cube dimension that corresponds to the relationship R_j. Finally, we define the function $ViewAtt(D^{VR_j}) = VA^{VR_j}$ that associates D^{VR_j} with an attribute belonging to one attribute hierarchy of VO^{VR_j}.

Example 4. Let us suppose that the user wants to analyze *max_rating* by *author* and by *year* and *top_keyword* by *author* and by *proceedings*. The user wants to display the publication titles, the authors' first names and surnames and the proceedings' titles. The view that corresponds to such an analysis is depicted in Fig. 3. Here, the VF features (the publication titles) are set to *Undefined* because the values of *max_rating* are aggregated at the year level along the hierarchy *H_time*. Formally, let C be the cube defined by this analysis context and V_RK the view that the user wants to display. Then, $VR_K = \Pi_V(C) = (F^{V\_RK}$, SM^{V_RK}, $SD^{V\_RK})$ where

- F^{V_RK}=Undefined;
- SM^{V_RK}={max_rating, top_keyword};
- $SD^{V\_RK} = \{Authors, Time, Conferences\}$ such that
  - $Authors = \{Author.Firstname, Author.Lastname\}$ where
    - ∗ *ViewObj(Authors) = Author*
    - ∗ $ViewAtt(Authors) = Author.Author\_id$
  - $Time = \{Date.Year\}$ where
    - ∗ $ViewObj(Time) = Date$
    - ∗ $ViewAtt(Time) = \{Date.Year\}$
  - $Conferences = \{Proceedings.Name\}$ where
    - ∗ $ViewObj(Conferences) = Proceedings$
    - ∗ $ViewAtt(Conferences) = Proceeding.Proceeding\_ID$

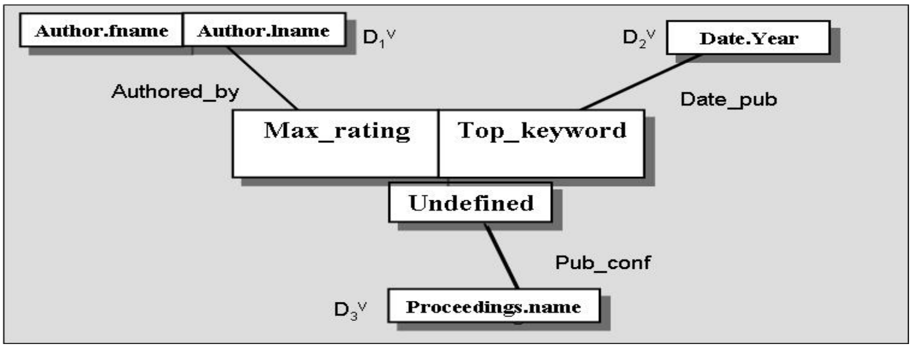


Fig. 3. Example of a view over a complex cube

Structuring the view and restricting the data. Structuring a view consists in adding or deleting features or measures according to the user's needs, i.e. to have more or less information displayed. The operations of restricting data are similar to *slice-and-dice* operations in traditional OLAP. They consist in applying a selection predicate either to the fact/dimension feature values or to the detailed/aggregated measure values. Conversely, unrestricting data consists in displaying all the values of a feature or a measure and it applies to previously restricted values. Table 3 summarizes these operations.

Table 3. Structure and data-related operators on views over complex cubes

Operation		Definition
Add / Remove measures		$ADD_{VM}(V^C, M_+) \| REM_{VM}(V^C, M_-)$
Add / Remove features	of fact	$ADD_{FF}(V^C, FF_+) \| REM_{FF}(V^C, FF_-)$
	of dimension	$ADD_{DF}(V^C, DF_+) \| REM_{DF}(V^C, DF_-)$
Restrict/Unrestrict data	of fact	$\sigma_{FF} V^C(P(FF)) \| \mu_{FF} V^C(FF)$
	of dimension	$\sigma_{DF} V^C(P(DF)) \| \mu_{DF} V^C(DF)$
	of measure	$\sigma_M V^C(P(M)) \| \mu_M V^C(M)$

Example 5. Examples of visualization operation on the view V_RK of example 4 are (1) adding the names of conferences: $ADD_{DF}(V\_RK$, Conferences.Conference.Name) and (2) displaying only publications of year 2000: $\sigma_{DF} V\_RK$(Time.Date.year = '2000').

3.3 Aggregate Operators

An aggregate operation consists in changing the associated object of a view dimension or its associated attribute. The measure values are then aggregated or detailed consequently. We define three kinds of aggregate operations.

- the *Object hierarchy-based* operations consist in changing the current AO by the object at the upper level according to an object hierarchy (rollup) or at the lower level (drill down);
- the *Attribute hierarchy-based* operations consist in changing the current AA by the attribute at the upper level according to an attribute hierarchy (rollup) or at the lower level (drill down);
- the *Hierarchyless operations* are applicable if AO (resp. AA) does not belong to any object-hierarchy (resp. to any attribute hierarchy). The hierarchyless rollup/drill-down consist respectively in removing/displaying the VD.

Note. Due to space limitation, we omit the formal notations for aggregate operations. Yet, we summarize them in table 4.

Table 4. Aggregate Operators

Operation		Definition
Rollup	Object hierarchy-based	$RollUp_{OH}(V^C, D^{VRRU}, OH^C)$
	Attribute hierarchy-based	$RollUp_{AH}(V^C, D^{VRRU}, AH^C)$
	Hierarchyless	$RollUp_{HL}(V^C, D^{VRRU})$
Drill down	Object hierarchy-based	$DrillDown_{OH}(V^C, D^{VRRU}, OH^C)$
	Attribute hierarchy-based	$DrillDown_{AH}(V^C, D^{VRRU}, AH^C)$
	Hierarchyless	$DrillDown_{HL}(V^C, D^{VRRU})$

Example 6. Examples of aggregate operations on the view *V_RK* are the following: (1) Attribute hierarchy-based rollup along the hierarchy *H_time* and according to the relationship *Date_pub* to get *max_rating* by *authors* and for all periods of *time*: $RollUp_{AH}(V\_RK, Time, H\_time)$. (2) Object hierarchy-based rollup along the hierarchy *H_conf* to get *top_keyword* by *author* and by *conference*: $RollUp_{OH}(V\_RK, Conferences, H\_Conf)$. (3) Hierarchyless rollup to get *max_rating* by *year* and *top_keyword* by *proceedings* for all *authors*: $RollUp_{HL}(V\_RK, Authors)$. The drill down operations are the converse of the previous operations.

4 Implementation

In order to validate our multidimensional schema and operators, we implemented the core of a warehousing and analysis framework (Fig. 4). We have translated the conceptual modeling elements into the logical and physical levels using XML. The choice of XML is motivated by its widespread use and its power to describe heterogeneous and diversely structured data or complex data in general. Thus, we developed an XML schema that describes the structure of any data warehouse and any complex cube. Then, for a functional validation, we used the dblp.xml file as the main source of our data warehouse. At the metadata level, we defined

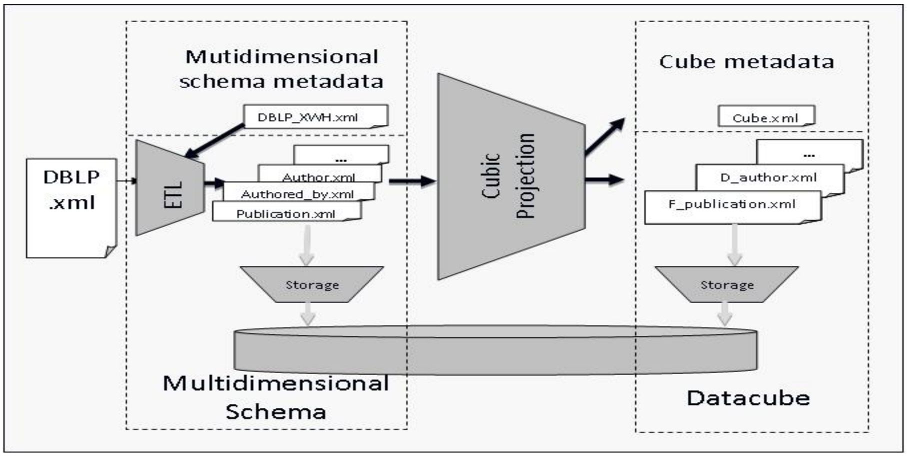


Fig. 4. System architecture

an XML file dblp_xwh.xml that describes the content of the data warehouse which is then materialized as a set of other XML files.

The modules of the platform the following. (1) the ETL module reads dblp.xml and loads the data into the data warehouse XML files. These files are then stored into a native XML database (eXist). (2) the cube specification module implements the cubic projection operator. It reads the meta data file (dblp_xwh.xml) as well as the data files and produces a metadata file (cube.xml) and a set of XML documents that contain the real data. The visualization and analysis-related operators will be implemented in a next step.

5 Conclusion

In this paper, we have presented a set of OLAP operators for a complex object-based multidimensional model. The first set of operators allows constructing of complex data cubes from the multidimensional schema or from existing cubes. The structure-related operators produce new cubes having a different structure than the original cubes whereas the data-related operators produce same-structured cubes but containing more or less data. We have also defined a projection operator over a data cube in order to display its data. Other view-related operators aim at displaying more or less features or measures on a view (structure-related operators) or at getting more or less data displayed (data-related operators). Finally, the aggregate operators aim at having more or less detail about the measures' values. There are many perspectives for our work. First, we plan to consider more complex-structured measures instead of simple attributes. Then, we shall extend the cube structure by considering the attribute hierarchies in relation to the fact and the object hierarchies that contain the fact. These hierarchies will then allow observing a same measure at different levels of the hierarchies and thus enable fact-based aggregate operations. Finally, we plan to extend the visualization and analysis operations to displaying the internal structure of the complex objects (e.g. the relationships between attributes) and thus enable relationship-aware analyzes.

References

1. Abelló, A., Samos, J., Saltor, F.: A framework for the classification and description of multidimensional data models. In: Mayr, H.C., Lazanský, J., Quirchmayr, G., Vogel, P. (eds.) DEXA 2001. LNCS, vol. 2113, pp. 668–677. Springer, Heidelberg (2001)
2. Bimonte, S., Tchounikine, A., Miquel, M.: Towards a spatial multidimensional model. In: Song, I.-Y., Trujillo, J. (eds.) Proceedings of the ACM 8th International Workshop on Data Warehousing and OLAP (DOLAP 2005), Bremen, Germany, pp. 39–46. ACM, New York (2005)
3. Boussaïd, O., Boukraâ, D.: Multidimensional Modeling of Complex Data. In: Encyclopedia of Data Warehousing and Mining, 2nd edn., pp. 1358–1364. IGI Publishing, Hershey (2009)
4. Boussaïd, O., Tanasescu, A., Bentayeb, F., Darmont, J.: Integration and dimensional modelling approaches for complex data warehousing. Journal of Global Optimization 37(4), 571–591 (2007)
5. Chaudhuri, S., Dayal, U.: An overview of data warehousing and olap technology. SIGMOD Record 26(1), 65–74 (1997)
6. Cuzzocrea, A., Mansmann, S.: OLAP Visualization: Models, Issues, and Techniques. In: Encyclopedia of Data Warehousing and Mining, 2nd edn., pp. 1439–1446. IGI Publishing, Hershey (2009)
7. Damiani, M.L., Spaccapietra, S.: Spatial Data Warehouse Modelling. In: Processing and Managing Complex Data for Decision Support. Idea Group Publishing, USA (2006)
8. Golfarelli, M., Rizzi, S., Vrdoljak, B.: Data warehouse design from XML sources. In: Proceedings of teh 4th ACM International Workshop on Data Warehousing and OLAP (DOLAP 2001), Atlanta, Georgia, USA (2001)
9. Inokuchi, A., Takeda, K.: A method for online analytical processing of text data. In: Silva, M.J., Laender, A.H.F., Baeza-Yates, R.A., McGuinness, D.L., Olstad, B., Olsen, Ø.H., Falcão, A.O. (eds.) Proceedings of the 16th ACM Conference on Information and Knowledge Management (CIKM 2007), Lisbon, Portuga, pp. 455–464. ACM, New York (2007)
10. Jensen, M.R., Møller, T.H., Pedersen, T.B.: Specifying olap cubes on XML data. Journal of Intelligent Information Systems 17(2-3), 255–280 (2001)
11. Lin, B., Hong, Y., Lee, Z.-H.: Data Warehouse Performance. In: Encyclopedia of Data Warehousing and Mining, 2nd edn., pp. 580–585. IGI Publishing, Hershey (2009)
12. Luján-Mora, S., Trujillo, J., Song, I.-Y.: Multidimensional modeling with UML package diagrams. In: Spaccapietra, S., March, S.T., Kambayashi, Y. (eds.) ER 2002. LNCS, vol. 2503, pp. 199–213. Springer, Heidelberg (2002)
13. Nassis, V., Rajugan, R., Dillon, T.S., Rahayu, J.W.: Conceptual design of XML document warehouses. In: Kambayashi, Y., Mohania, M., Wöß, W. (eds.) DaWaK 2004. LNCS, vol. 3181, pp. 1–14. Springer, Heidelberg (2004)

14. Pedersen, T.B., Gu, J., Shoshani, A., Jensen, C.S.: Object-extended olap querying. Data Knowledge Engineering 68(5), 453–480 (2009)
15. Ravat, F., Teste, O., Tournier, R., Zurfluh, G.: Algebraic and graphic languages for olap manipulations. International Journal of Data Warehousing and Mining 4(1), 17–46 (2008)
16. Vaisman, A.A., Mendelzon, A.O.: A temporal query language for olap: Implementation and a case study. In: Ghelli, G., Grahne, G. (eds.) DBPL 2001. LNCS, vol. 2397, pp. 78–96. Springer, Heidelberg (2002)
17. Wiwatwattana, N., Jagadish, H.V., Lakshmanan, L.V.S., Srivastava, D.: X^3: A cube operator for XML olap. In: Proceedings of the 23rd International Conference on Data Engineering (ICDE 2007), Istanbul, Turkey, pp. 916–925. IEEE, Los Alamitos (2007)

ADBdesign: An Approach to Automated Initial Conceptual Database Design Based on Business Activity Diagrams

Drazen Brdjanin[1], Slavko Maric[1], and Dejan Gunjic[2]

[1] University of Banja Luka, Faculty of Electrical Engineering, Patre 5, 78000 Banja Luka, Bosnia and Herzegovina
{bdrazen,ms}@etfbl.net
[2] NITES, Petra Kocica 41, 78000 Banja Luka, Bosnia and Herzegovina
dejan.gunjic@gmail.com

Abstract. This paper presents a new approach to automated initial conceptual database design based on detailed UML business activity diagrams. The most important concepts of detailed business activity diagrams, as a frequently used business process modeling notation, are identified and the XMI represented. Based on those concepts, we define the rules for the automated generation of the class diagram as the target initial conceptual database model. We also give a short description of the used software development environment and implemented generator with some experimental results of application to a real business model.

Keywords: business activity diagram, class diagram, database, initial conceptual model, UML.

1 Introduction

Business systems are complex systems that provide required products and/or services to customers. They are characterised by an appropriate organisational structure and business processes taking place in order to satisfy customers' needs or requirements. Customers, business processes and the organisational structure together make up what is often called business domain. Modeling of business domain is the subject of the business modeling discipline. The business model, as the result of business modeling, is an abstraction of business system elements and their interrelationships [1].

According to modern methodologies, business modeling is the first phase of software system development for two reasons. On the one hand, the business model is used for the identification of system requirements, i.e. the architecture of an information system that best supports a given business system. On the other hand, through some mappings and/or transformations, business models can be used directly for building target software system models, which significantly increases the software development efficiency. This aspect is the very foundation of *Model Driven Development* (MDD), a paradigm that sees the business model (CIM - *Computational Independent Model*) as the basis for automated *Platform Independent Model* (PIM) design [2].

The related literature suggests a large number of database design methodologies and notations [3]. Database design mainly undergoes the following phases: requirements analysis, conceptual design, logical design and physical design. Each phase ends in an appropriate model, which is characterised by the presented abstraction level on the one hand, and the reached level of the implementation details of the target database on the other. The main goal of conceptual design is a model that provides an overall view of information in the entire system. This model is usually called the conceptual database model and represents a semantic data model. The related literature is more significantly focused on conceptual design than on other phases, considering it to be the most important design phase, because the following phases are only subsequent transformations of the model from the previous one.

The initial phase in database design includes the preliminary specification and modeling of the information needs of the business system for which a database is designed. Use cases, as one of important approaches in software engineering [4], are a frequently used mechanism for the identification of system functionalities and a reliable basis for the system requirements specification at a high level of abstraction. It is business use case driven business models that are used in this paper as the starting point for the automated design of the initial conceptual model of relational database.

Modern database design approaches [5] emphasize the initial phase, considering that the most important part of conceptual design is already completed in that phase, because all business entities and their interrelationships are mainly identified, so the following design phases are mainly just subsequent transformations of the initial conceptual model. Therefore, the database design process consists of the following phases: initial conceptual design, conceptual model refinement and optimization, model mapping to the target database schema, and schema optimization. During this process, the conceptual model undergoes several transformations which either maintain the equivalence of the model or change it by introducing new or removing some existing concepts.

The fact that business modeling and database design use different notations that usually don't conform to the same or common metamodel imposes a particular problem and challenge in automated initial conceptual model design based on the business model. Business modeling is mainly characterised by process-oriented notations, such as: IDEF0 [6], EPC [7], Petri nets [8], BPMN [9], etc. On the other hand, the E-R (*Entity-Relationship*) notation [10] or one of its modifications such as IE [11] is traditionally used for conceptual database modeling, while the development of UML (*Unified Modeling Language*) has seen an increasing use of the UML class diagram as well [12]. Different, i.e. non-harmonized notations are one of the reasons for a fairly small number of papers in the field.

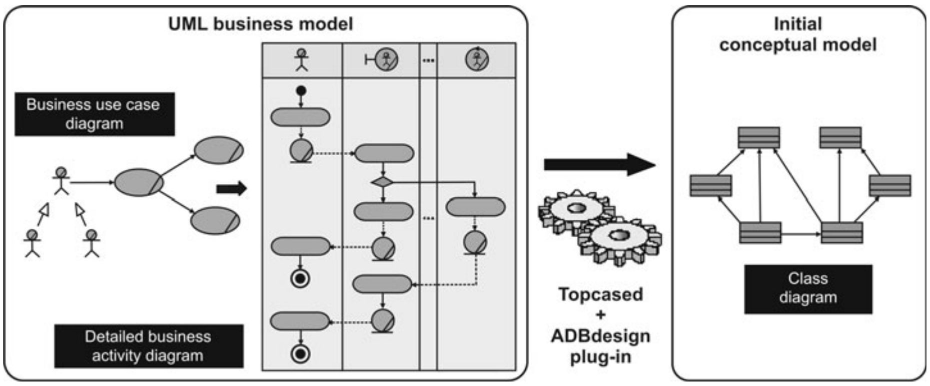

UML business model
Business use case diagram
Detailed business activity diagram
Topcased + ADBdesign plug-in
Initial conceptual model
Class diagram

Fig. 1. Approach to automated initial conceptual model design

One of the ways to overcome the problem of non-harmonised notations is the use of UML for both business modeling and for database design. UML [13] is the industry standard for software modeling, but the open concept, which allows the specialisation of the rich notation, ensures the application of UML in other, non-software disciplines as well. This paper presents an approach to automated initial conceptual database model design which uses a unified notation (Fig. 1). The proposed approach aims to generate the UML class diagram as the target initial conceptual model, on the basis of the detailed UML business activity diagram as the diagram which documents business use cases [14].

Following the introduction, the second section of the paper describes detailed business activity diagrams as the basis for automated initial conceptual model design. The third section presents the XMI representation of detailed activity diagrams as the source diagrams, and of class diagrams as the target diagrams. Target generator requirements and implementation are described in the fourth section, while the experimental results of the generator's application in a concrete business system are given in the fifth section. An overview of the related work is presented in the sixth section. Finally, we conclude the paper and highlight the directions for further research.

2 Detailed Business Activity Diagrams

The UML activity diagram is a widely accepted business process modeling notation [15]. There are a large number of papers dealing with the formalisation of semantics and the analysis of activity diagram suitability for business modeling [16].

In business modeling, the activity diagram can be used in two ways [17]. The first form is the so-called *macroactivity diagram*, which is used in developing the initial business model for the rough specification of activities which realise business use cases, i.e. for modeling business processes at a high level of abstraction.

The second form is the so-called *detailed activity diagram* which is developed by populating the macroactivity diagram with all business process participants (business actors, workers, organisational units, etc.), used objects and object flows. The participants' areas of responsibility are modelled by swimlanes. All activities, objects and object flows related to a participant are shown in the same swimlane. Hence, there are as many swimlanes as there are participants involved in the process. The input of an object into an activity is an input object flow, which is modelled as a dependency directed from the object toward the activity. An object which is an output from an activity is connected by a dependency directed from the activity toward the object, which is called output object flow.

Detailed activity diagrams represent a reliable starting point for developing the initial conceptual model, because business objects, object flows, activities on objects, and participants' responsibilities fully document the identified business use cases. It is these details (participants and objects) that are the basis for the initial conceptual model, because they are directly mapped into the classes of the target class diagram, while the identified activities are mapped into the class associations with the appropriate multiplicity, as given in [14] (Fig. 2).

The relationships between all participants (be it the business actor, worker or organisational unit) and business objects are *agent-object* relationships [18] (the business actor buys a product or files an application, the worker issues a required document, etc.) Multiplicity is most often that of type "1..*" (the business actor can file several applications, buy several products, etc.). Other

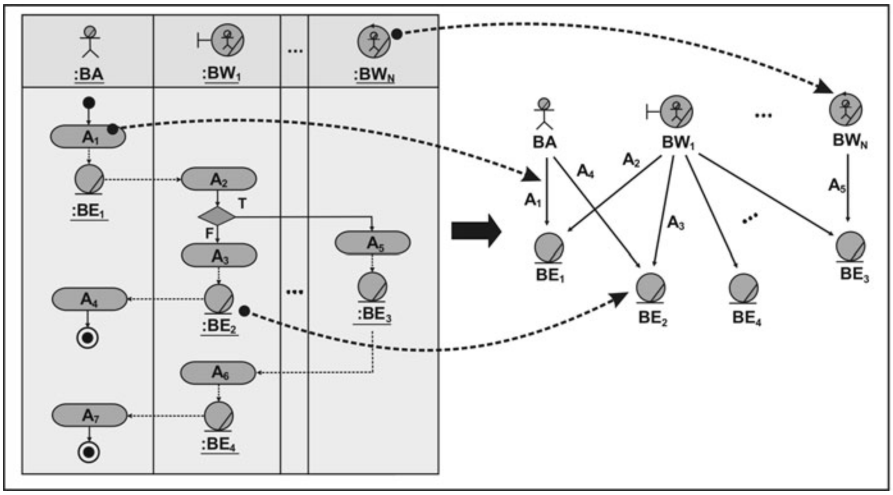


Fig. 2. Mapping of detailed activity diagram into (business) class diagram

types of multiplicity can often be classified under type "1..*" multiplicity (e.g. "1..1" is a special case of "1..*" multiplicity).

3 XMI Representation of UML Diagrams

XMI (*XML Metadata Interchange*) [19] is the OMG standard for platform independent metadata interchange enabling the serialization of MOF (*Meta-Object Facility*)-based models and metamodels into XML, and vice versa, the visualisation of MOF-based models and metamodels from XML.

3.1 XMI Representation of Detailed Activity Diagram

According to [19], each MOF instance, and in this case each element of the detailed activity diagram, is represented by an appropriate XML element marked by a specific XML tag `<group>`, `<node>` or `<edge>` and contains a certain number of attributes. Each XML element is at least characterised by the `xmi:type` attribute as the UML metaclass identifier, and the `xmi:id` attribute as the unique instance identifier in the model. Apart from these, there may be other attributes as well, such as: `name` for naming an instance, `inPartition` for identifying the swimlane an instance belongs to, `incoming` and `outgoing` for identifying input and output object flows, etc.

For illustration, Table 1 shows the XMI representation of the key concepts of the generic activity diagram in Fig. 3 (left).

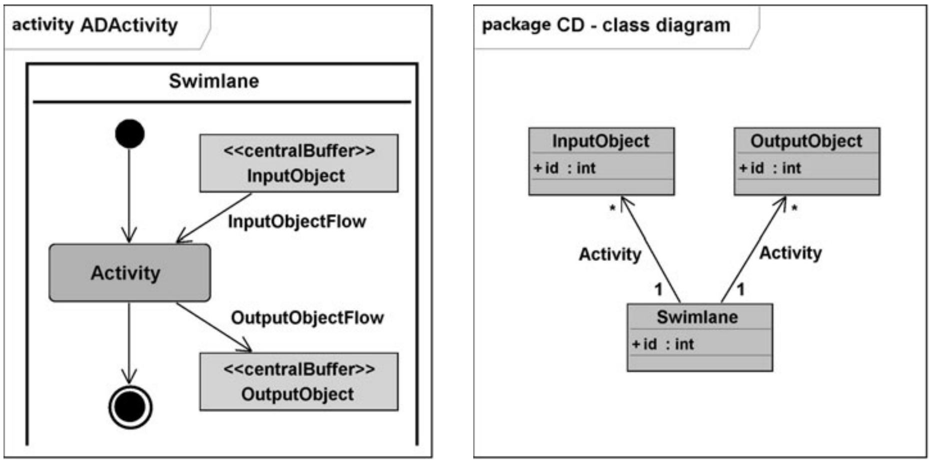


Fig. 3. Generic activity diagram *(left)* and corresponding class diagram *(right)*

Table 1. XMI representation of activity diagram key concepts

Element	**XMI representation**
`Swimlane`	`<group xmi:type="uml:ActivityPartition" xmi:id="id_P" name="Swimlane" node="id_A id_IO id_OO ..." edge="id_IF id_OF ..."/>`
`Activity`	`<node xmi:type="uml:OpaqueAction" xmi:id="id_A" name="Activity" inPartition="id_P" outgoing="id_OF ..." incoming="id_IF ..."/>`
`InputObject`	`<node xmi:type="uml:CentralBufferNode" xmi:id="id_IO" name="InputObject" outgoing="id_IF" inPartition="id_P"/>`
`OutputObject`	`<node xmi:type="uml:CentralBufferNode" xmi:id="id_OO" name="OutputObject" incoming="id_OF" inPartition="id_P"/>`
`InputObjectFlow`	`<edge xmi:type="uml:ObjectFlow" xmi:id="id_IF" name="InputObjectFlow" source="id_IO" target="id_A" inPartition="id_P"/>`
`OutputObjectFlow`	`<edge xmi:type="uml:ObjectFlow" xmi:id="id_OF" name="OutputObjectFlow" source="id_A" target="id_OO" inPartition="id_P"/>`

3.2 XMI Representation of Class Diagram

According to [19], each class in the class diagram is represented by an appropriate XML element which is marked by the XML tag `<packagedElement>` and whose `xmi:type` attribute has the `"uml:Class"` value.

Each class attribute is represented by an XML element marked by the tag `<ownedAtttribute>`. Attributes may be class data members, as well as those based on associations with other classes. Apart from the unique identifier `xmi:id`, an XML element representing a class data member has two attributes - `name` and `type`. The attribute `name` contains the name of a class data member (e.g. `firstName`), while the attribute `type` represents the data member type.

An XML element representing an attribute related to an association with another class is always called `"target"`, and apart from the `xmi:id`, it has the attributes `association` and `type`, which contain the identifiers of an association and the other class with which the association has been established, respectively. Attributes related to associations are additionally characterised by the target end multiplicity (`<upperValue>` and `<lowerValue>`).

Each class association is represented by an XML element marked by the `<packagedElement>` tag, whose `xmi:type` attribute has the `"uml:Association"` value, while two `memberEnd` attributes identify the `source` and `target` association ends. An association is also characterised by the `<ownedEnd>` element which defines the source multiplicity (the target multiplicity is already defined).

Table 2. XMI representation of class diagram key concepts

Element	**XMI representation**

class `InputObject`	`<packagedElement xmi:type="uml:Class"`
	`    xmi:id="id_IO" name="InputObject">`
	`  <ownedAttribute xmi:id="id_aO" name="id" type="int"/>`
	`</packagedElement>`

class `Swimlane`	`<packagedElement xmi:type="uml:Class"`
	`    xmi:id="id_P" name="Swimlane">`
	`  <ownedAttribute xmi:id="id_aP" name="id" type="int"/>`
	`  <ownedAttribute xmi:id="id_t" name="target"`
	`      type="id_IO" association="id_A">`
	`    <upperValue xmi:type="uml:LiteralUnlimitedNatural"`
	`      xmi:id="id_uP" value="*"/>`
	`    <lowerValue xmi:type="uml:LiteralInteger"`
	`      xmi:id="id_lP"/>`
	`  </ownedAttribute>`
	`</packagedElement>`

association `Activity`	`<packagedElement xmi:type="uml:Association" xmi:id="id_A"`
	`    name="Activity" memberEnd="id_t id_oEA">`
	`  <ownedEnd xmi:id="id_oEA" name="source"`
	`      type="id_P" association="id_A">`
	`    <upperValue xmi:type="uml:LiteralUnlimitedNatural"`
	`      xmi:id="id_uO" value="1"/>`
	`    <lowerValue xmi:type="uml:LiteralInteger"`
	`      xmi:id="id_lO" value="1"/>`
	`  </ownedEnd>`
	`</packagedElement>`

For illustration, Table 2 shows the XMI representation of the key concepts of the generic class diagram in Fig. 3 (right), which is the initial conceptual model for the given activity diagram (`OutputObject` class is omitted, because its representation is very similar to the `InputObject` class).

4 Initial Conceptual Model Generator

This section defines the requirements for the target generator of the initial conceptual model and describes its implementation according to those requirements.

4.1 Requirements for Target Generator

The initial conceptual model generator is to implement the transformation process illustrated by the activity diagram in Fig. 4. Following the selection of a source activity diagram, an appropriate XML tree is to be generated in order to enable the extraction of characteristic concepts and the generation of the target class diagram.

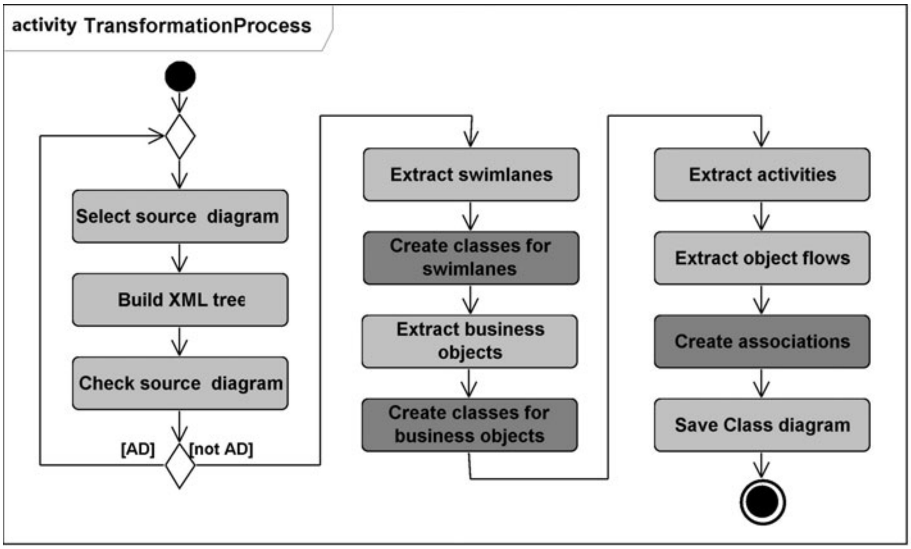


Fig. 4. Initial conceptual model design process

The first step in creating a class diagram is the extraction of swimlanes from the XML tree and the generation of the corresponding classes. The generator is to recognise a swimlane as the `<group>` element, whose `xmi:type` attribute has the `"uml:ActivityPartition"` value as shown in Table 1. For each swimlane a class of the same name is to be created in accordance with Table 2. Additionally, the attribute `"id"` representing the primary key in the target key-based initial conceptual database model, is to be added to the created class. Following the same analogy, business objects are to be extracted from the XML tree and the classes of the same name are to be generated, in accordance with the XMI definitions provided in Tables 1 and 2.

After generating classes for the swimlanes and business objects, the associations of the created classes are to be generated based on object flows and activities. The generator is to extract object flows (according to the Table 1), and to identify the source and target for each object flow in order to distinguish input and output object flows, since this directly affects the association multiplicity. The source end of an association is always a class representing a swimlane. Its identifier can be determined indirectly. Firstly, the identifier of an activity for which the given object flow represents an input or output is to be determined, and then the swimlane the activity belongs to, is to be identified. The target end is always a class representing a business object and its identifier is determined directly. An association is to be generated in accordance with the XMI definition given in Table 2, with multiplicity "1" on the source end, and "*" on the target end. The name of the association corresponds to the activity performed on the object.

4.2 Implementation

The target generator can be implemented in different ways and on different development platforms. Given the experimental nature of the approach and efforts to implement the generator independently from commercial tools, the choice is reduced to an Eclipse-based development environment.

Although there are a relatively large number of available Eclipse plugins enabling UML model manipulation, the number of plugins enabling an adequate manipulation of detailed activity diagrams, as well as a satisfactory visualisation of the generated class diagram, is relatively small. This paper uses the Eclipse-based Topcased development platform [20], v3.2.0.

Topcased. Topcased v3.2.0 (*Toolkit in Open-source for Critical Application & Systems Development*) is a system/software engineering toolset based on Galileo (Eclipse 3.5). It includes several graphical model editors (UML, SysML, SAM, AADL), OCL tools, QVT processor, some code generators (UML2Java, UML2Python, etc), and some other experimental features. Topcased UML toolkit enables the manipulation of most UML diagrams and ensures a satisfactory visualisation of automatically generated diagrams.

The functionality of the Topcased platform and the implemented initial conceptual database model generator is based on the standard UML2 Eclipse plugin as the EMF (*Eclipse Modeling Framework*) - based [21] implementation of the UML 2.1 specification [22] for the Eclipse platform, which enables visual UML modeling, as well as the program generation of UML diagrams.

Each UML diagram in the Topcased model is represented by two files of the same name, but different extensions. The *.uml* file contains the XMI representation of a diagram, while the *.umldi* file describes the visualisation of a diagram. The implemented generator processes the *.uml* description of the activity diagram and generates the *.uml* representation of the class diagram, and the visualisation (generation of the corresponding *.umldi* file) is then performed by using the integrated Topcased functionality. Given the applied UML2 plugin, the generator manipulates diagrams that are compliant with the UML 2.1.0 and the XMI 2.1 specifications.

ADBdesign plugin. The proposed activity diagram transformations are of endogenous nature, because both the source and the target diagram have the same metamodel, and the transformation rules are relatively simple. The generator implementation is therefore not based on the QVT [23] approach, but on the combination of the DOM XML parser for the source diagram analysis and the UML2 factory for the target diagram generation [24].

The ADBdesign plugin implements the generator in accordance with the identified requirements. The architecture of the implemented plugin is shown in Fig. 5. The `GUI` package contains classes that enable integration into the Topcased environment and user interface, while the `Generator` package contains the `ad2cd` class, which implements the transformation process.

Following the selection of a source diagram (`ad2cdWizard`), an appropriate XML tree is generated by `fileToDocument()` method (based on the DOM

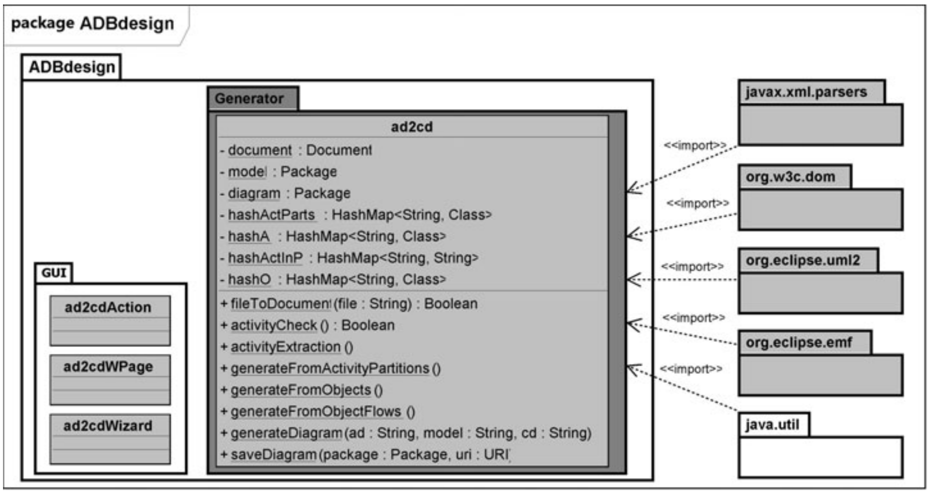

package ADBdesign
ADBdesign
Generator
ad2cd
- document : Document
- model : Package
- diagram : Package
- hashActParts : HashMap<String, Class>
- hashA : HashMap<String, Class>
- hashActInP : HashMap<String, String>
- hashO : HashMap<String, Class>
+ fileToDocument (file : String) : Boolean
+ activityCheck () : Boolean
+ activityExtraction ()
+ generateFromActivityPartitions ()
+ generateFromObjects ()
+ generateFromObjectFlows ()
+ generateDiagram (ad : String, model : String, cd : String)
+ saveDiagram (package : Package, uri : URI)
GUI
ad2cdAction
ad2cdWPage
ad2cdWizard
<<import>>
javax.xml.parsers
org.w3c.dom
org.eclipse.uml2
org.eclipse.emf
java.util

Fig. 5. ADBdesign plugin architecture

`DocumentBuilderFactory`). The generation of the target class diagram starts if the selected diagram is an activity diagram (method `activityCheck()`).

The extraction of swimlanes from the XML tree and the generation of appropriate classes is done by `generateFromActivityPartitions()` method. The XMI identification of the created classes is done by the used UML2 generator of the Topcased development platform. The attribute `"id"`, which is the primary key, is added to the generated classes. The code of the described generator is provided in the following listing:

```
public static void generateFromActivityPartitions()
{
  org.eclipse.uml2.uml.Class classAP;
  NodeList swlanes = document.getElementsByTagName("group");
  for (int i=0; i<swlanes.getLength(); i++)
  {
    NamedNodeMap attribs = swlanes.item(i).getAttributes();
    Attr typeA = (Attr)attribs.getNamedItem("xmi:type");
    String type = typeA.getValue().toString();
    if (type.equals("uml:ActivityPartition"))
    {
      Attr attribute = (Attr)attribs.getNamedItem("name");
      String name = attribute.getValue().toString();
      classAP = diagram.createOwnedClass(name, false);
      classAP.createOwnedAttribute("id", primitiveType_int);
    }
  }
}
```

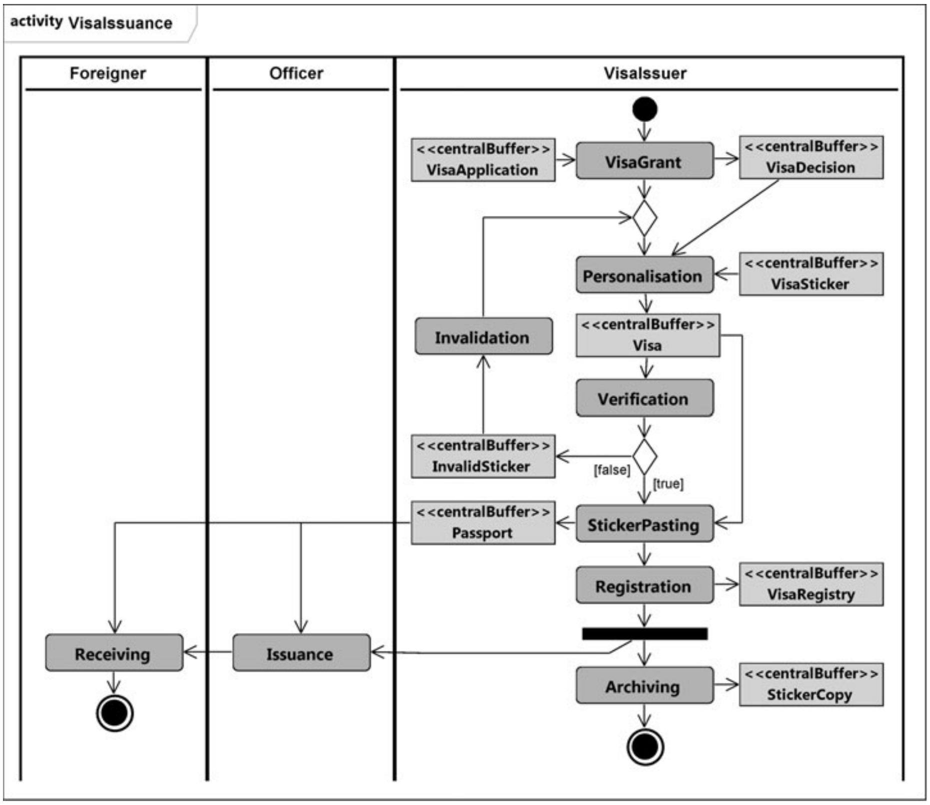
The `generateFromObjects()` generator, which extracts business objects and generates appropriate classes, is implemented following the same analogy. Given the similarity, we don't describe the implementation.

The `generateFromObjectFlows()` method implements the associations generator. It uses the `HashMap<String,Class>` objects `hashO` and `hashA` with data on the created classes for business objects and extracted activities respectively, as well as the `HashMap<String,String>` object `hashActInP` with data on activities belonging to the swimlanes. These objects are created by `activityExtraction()` method. The code of the described generator is provided in the following listing:

```
public static void generateFromObjectFlows()
{
  NodeList edges = document.getElementsByTagName("edge");
  for (int i=0; i<edges.getLength(); i++)
  {
    NamedNodeMap attribs = edges.item(i).getAttributes();
    Attr typeA = (Attr)attribs.getNamedItem("xmi:type");
    String type = typeA.getValue().toString();
    if (type.equals("uml:ObjectFlow"))
    {
      Attr idSoA = (Attr)attribs.getNamedItem("source");
      String idS = idSoA.getValue().toString();
      Attr idTaA = (Attr)attribs.getNamedItem("target");
      String idT = idTaA.getValue().toString();
      String idA = (hashA.containsKey(idS)) ? idS : idT;
      String idO = (hashA.containsKey(idS)) ? idT : idS;
      if (hashActInP.get(idA) != null)
      {
        String source = hashActInP.get(idA);
        String name = hashA.get(idA);
        org.eclipse.uml2.uml.Class classO =
             (org.eclipse.uml2.uml.Class)hashO.get(idO);
        org.eclipse.uml2.uml.Class classAP =
             (org.eclipse.uml2.uml.Class)hashActParts.get(source);
        Association asc = classAP.createAssociation(true,
             AggregationKind.NONE_LITERAL, "target", 0,
             LiteralUnlimitedNatural.UNLIMITED, classO, false,
             AggregationKind.NONE_LITERAL, "source", 1, 1);
        asc.setName(name);
      }
    }
  }
}
```

5 "Real-World" Example

The implemented generator of the initial conceptual database model is applied to the business model of the visa issuance system, which is also used for illustration


activity VisaIssuance
Foreigner
Officer
VisaIssuer
<<centralBuffer>> VisaApplication
VisaGrant
<<centralBuffer>> VisaDecision
Personalisation
<<centralBuffer>> VisaSticker
Invalidation
<<centralBuffer>> Visa
Verification
<<centralBuffer>> InvalidSticker
[false]
[true]
<<centralBuffer>> Passport
StickerPasting
Registration
<<centralBuffer>> VisaRegistry
Receiving
Issuance
Archiving
<<centralBuffer>> StickerCopy

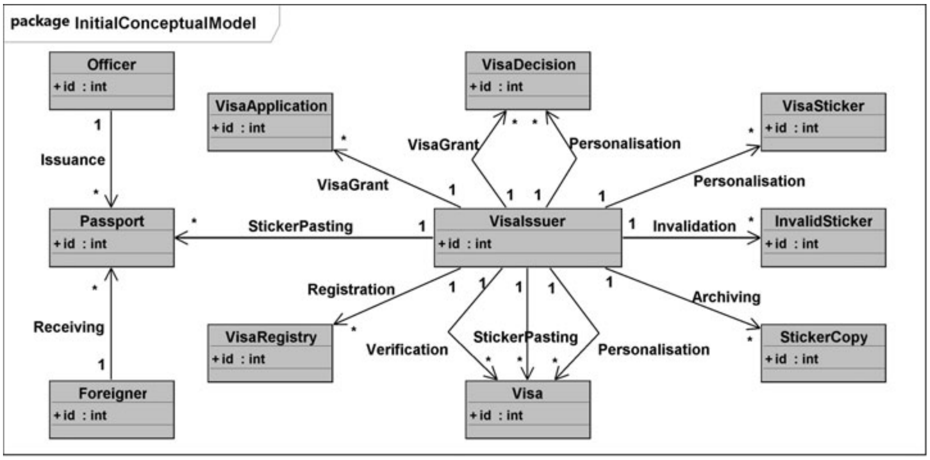


Fig. 6. Detailed activity diagram for `VisaIssuance` business use case *(up)* and corresponding automatically generated initial conceptual model *(down)*

in [14]. Here we take the `VisaIssuance` business use case for illustration. A detailed activity diagram, which documents the realization of the selected business use case, is provided in Fig. 6 (up). Since the presented workflow is very obvious, we don't provide its description.

The result of the generator's application to the *.uml* file containing the XMI representation of the taken detailed activity diagram is the *.uml* file that contains the XMI representation of the target class diagram. Due to its size, we omit the listing, but we provide the visualisation result in the Topcased environment (Fig. 6 (down)). It is obvious that classes named `Foreigner`, `Officer` and `VisaIssuer` are generated directly from swimlanes, while other classes are generated directly from used business objects. Class associations are generated and named according to activities performed on those business objects by appropriate business process participant. Beside the single class associations, the implemented generator is also able to generate multiple associations between classes, such as associations named `Personalisation`, `StickerPasting` and `Verification` that are established between `Visa` and `VisaIssuer` classes.

6 Related Work

Although the idea of the conceptual database model design based on the business model is not very new, there are no papers that present implemented automatic generator and provide experimental results, while the great majority just give the method overview.

Kamimura *et al.* in [25] propose detailed algorithm for generating the E-R diagram using the *well-disciplined* IDEF0-based business model, but without an actual implementation.

Garcia Molina *et al.* in [26] propose an approach to the transition from business models to the initial conceptual model, which is based on detailed UML activity diagrams and supplementary glossary. They propose the direct mapping of all information objects in activity diagram to the respective classes in the target class diagram, but don't propose creation of classes for business process participants. They base creation of class associations on business rules informally specified in glossary, which is not suitable basis for automatic generation.

Suarez *et al.* in [27] follow the approach proposed by Garcia Molina *et al.* and propose the enhancement in creation of class associations. They propose creation of associations for activities that have input and output objects, as direct mapping of those activities to the respective associations between classes that correspond to input and output objects. This proposal can be used for automated generation of associations, but has limitations related to automated generation of association multiplicity, since they don't propose the explicit rules.

In this paper, we follow the approach proposed by Brdjanin and Maric in [14]. Beside the proposal for direct mapping of all business objects into the respective classes, they propose direct mapping for all business process participants, too. They define rules for creation of associations between business process participants and business objects based on activities performed on those objects. Some further transformations of the initial conceptual model are also specified and illustrated, but without an actual implementation, too.

7 Conclusion and Future Work

Business modeling and database design use different notations that usually don't conform to the same or common metamodel. That fact imposes a particular problem and challenge in automated design of the initial conceptual database model based on the business model. Another reason for a fairly small number of papers in the field, beside the different and/or non-harmonized notations, is related to the semantic capacity of business models, which has not been completely explored yet, and should be exploited for automated conceptual database model design.

This paper presents an original and unique approach to the automated design of the initial conceptual key-based model of the relational database which is based on detailed activity diagrams. The described approach and implemented generator proved the concept and imply that detailed business activity diagrams (as a frequently used business process modeling notation) have a semantic capacity for automated initial conceptual model design.

The proposed approach can significantly speed up design of the database conceptual model, especially for business models that contain large number of business use cases. Then, the automatic transformation and generation process should be applied to all detailed activity diagrams in the business model. In that case, the generator incrementally builds the initial conceptual model by incremental addition of new classes and associations for each activity diagram.

The implemented automatic generator of the class diagram (as the initial conceptual model) is the main advantage of this approach as compared to the existing few approaches that take business models as a direct basis for database design. Given the fact that the shortcomings and limitations of the proposed approach mainly result from the insufficiently explored potential of activity diagrams, further research will focus on the full identification and additional enhancement of the semantic capacity of detailed activity diagrams for automated database design, the complete definition of the transformation rules and the formalization of the approach.

References

1. Eriksson, H., Penker, M.: Business Modeling with UML. OMG Press, New York (2000)
2. Rodriguez, A., Fernandez-Medina, E., Piattini, M.: CIM to PIM Transformation: A Reality. In: Xu, L., Tjoa, A., Chaudry, S. (eds.) Research and Practical Issues of Enterprise Information Systems II, vol. 2, pp. 1239–1249. Springer, Boston (2008)
3. Elmasri, R., Navathe, S.: Fundamentals of Database Systems, 5th edn. Addison-Wesley, Reading (2006)
4. Jacobson, I.: Object-Oriented Software Engineering. Addison-Wesley, Reading (1992)

5. Proper, H., Halpin, T.: Conceptual Schema Optimisation - Database Optimisation before sliding down the Waterfall. DoCS, University of Queensland (2004)
6. National Institute of Standards and Technology (NIST): FIPSP 183 - Integration Definition for Function Modeling (IDEF0). NIST, Gaithersburg (1993)
7. Scheer, A.: Business Process Engineering: Reference Models for Industrial Enterprises, 2nd edn. Springer, New York (1994)
8. Reising, W., Muchnick, S., Schnupp, P.: A Primer in Petri Net Design. Springer, New York (1992)
9. White, S., Miers, D.: BPMN Modeling and Reference Guide. Future Strategies, Lighthouse Point (2008)
10. Chen, P.: The Entity-Relationship Model: Toward a Unified View of Data. ACM ToDS 1(1), 9–36 (1976)
11. Martin, J.: Information Engineering. Prentice Hall, Englewood Cliffs (1990)
12. Naiburg, E., Maksimchuk, R.: UML for Database Design. Addison-Wesley, Reading (2001)
13. Booch, G., Rumbaugh, J., Jacobson, I.: The Unified Modeling Language User Guide, 2nd edn. Addison-Wesley Professional, Reading (2005)
14. Brdjanin, D., Maric, S.: An Example of Use-Case-driven Conceptual Design of Relational Database. In: EUROCON 2007 - The Int. Conference on Computer as a Tool, pp. 538–545. IEEE Press, New York (2007)
15. Ko, R., Lee, S., Lee, E.: Business process management (BPM) standards: A survey. Business Process Management Journal 15(5), 744–791 (2009)
16. Russell, N., van der Aalst, W., ter Hofstede, A., Wohed, P.: On the Suitability of UML 2.0 Activity Diagrams for Business Process Modeling. In: 3rd Asia-Pacific Conference on Conceptual Modeling, pp. 95–104. Australian Computer Society, Darlinghurst (2006)
17. Brdjanin, D., Maric, S.: UML-business profile-based Business Modeling in Iterative-Incremental Software Development. In: EUROCON 2005 - The Int. Conference on Computer as a Tool, pp. 1263–1266. IEEE Press, New York (2005)
18. Storey, V.: Understanding Semantic Relationships. VLDB Journal 2(4), 455–488 (1993)
19. Object Management Group (OMG): MOF2.0/XMI Mapping, v 2.1.1. OMG (2007)
20. TOPCASED Project: Toolkit in Open-source for Critical Application & SystEms Development, v 3.2.0, http://www.topcased.org
21. Budinsky, F., Steinberg, D., Merks, E., Ellersick, R., Grose, T.: Eclipse Modeling Framework. Pearson Education, Boston (2003)
22. Object Management Group (OMG): Unified Modeling Language: Superstructure, v 2.1.1. OMG (2007)
23. Object Management Group (OMG): MOF 2.0 Query / View / Transformation Specification, v 1.0. OMG (2008)
24. Hussey, K.: Getting Started with UML2. IBM Corp., New York (2006)
25. Kamimura, M., Inoue, K., Hasegawa, A., Kawabata, R., Kumagai, S., Itoh, K.: Integrated Diagrammatic Representations For Data Design In Collaborative Processes. Journal of Integrated Design & Process Science 7(4), 35–49 (2003)
26. Garcia Molina, J., Jose Ortin, M., Moros, B., Nicolas, J., Toval, A.: Towards Use Case and Conceptual Models through Business Modeling. In: Laender, A.H.F., Liddle, S.W., Storey, V.C. (eds.) ER 2000. LNCS, vol. 1920, pp. 281–294. Springer, Heidelberg (2000)
27. Suarez, E., Delgado, M., Vidal, E.: Transformation of a Process Business Model to Domain Model. In: WCE 2008 - World Congress on Engineering, vol. 1, pp. 165–169. IAENG, London (2008)

Efficiently Computing and Querying Multidimensional OLAP Data Cubes over Probabilistic Relational Data

Alfredo Cuzzocrea[1] and Dimitrios Gunopulos[2]

[1] ICAR-CNR and University of Calabria, Italy
[2] Dept. of Informatics and Telecommunications
University of Athens, Greece
cuzzocrea@si.deis.unical.it, dg@di.uoa.gr

Abstract. Focusing on novel database application scenarios, where datasets arise more and more in uncertain and imprecise formats, in this paper we propose a novel framework for efficiently computing and querying multidimensional OLAP data cubes over probabilistic data, which well-capture previous kinds of data. Several models and algorithms supported in our proposed framework are formally presented and described in details, based on well-understood theoretical statistical/probabilistic tools, which converge to the definition of the so-called probabilistic OLAP data cubes, the most prominent result of our research. Finally, we complete our analytical contribution by introducing an innovative Probability Distribution Function (PDF)-based approach for efficiently querying probabilistic OLAP data cubes.

1 Introduction

Multidimensional OLAP data cubes [20] are powerful tools allowing us to support rich and multi-perspective analysis over large amounts of data sets, based on a multi-dimensional and multi-resolution vision of data. Efficiently computing OLAP data cubes over the input dataset (e.g., relational databases) is a well-known research challenge that has been deeply investigated during last decades (e.g., [22,1]), with alternate fortune. *Probabilistic data* (e.g., [3,10,16,17,32,4,2,34]) are becoming one of the most attracting kinds of data for database researchers, due to the fact such a format/formalism perfectly captures two novel, interesting classes of datasets that very often occur in modern database application scenarios, namely *uncertain* and *imprecise data*. Uncertain and imprecise data are indeed very popular, as uncertainty and imprecision affect the same processes devoted to collect data from input data sources and make use of these data in order to populate the target database. Consider, for instance, the simplest case represented by a *sensory database* [5] populated by a sensor network monitoring the temperature T of a given geographic area S. Here, being T monitoring a natural, real-life measure, it is likely to retrieve an uncertain and imprecise *estimate* of T, denoted by $\tilde{T}$, with a given *confidence interval* [31], denoted by $[\tilde{T}_{min}, \tilde{T}_{max}]$, such that $\tilde{T}_{min} < \tilde{T}_{max}$, having a certain probability p_T, such that $0 \leq p_T \leq 1$, rather than to obtain the *exact value* of T, denoted by $\hat{T}$. The semantics of this confidence-interval-based model states that the (estimated) value of T, $\tilde{T}$, ranges between $\tilde{T}_{min}$ and $\tilde{T}_{max}$ with probability p_T. In popular probabilistic database models (e.g., [4,2,34]), confidence intervals and related probabilities are directly embedded into the *probabilistic tables* directly, thus originating *probabilistic attributes* storing *probabilistic* (*attribute*) *values*, which compose *probabilistic tuples*. Physical reasons of uncertain and imprecision of data are many-fold, and they can be found in inherent randomness and incompleteness of data, sampling errors, human errors, instrument errors, data unavailability, delayed data updates, and so forth.

Since probabilistic datasets are becoming very popular, it is natural and reasonable to define and introduce the problem of *efficiently computing OLAP data cubes over probabilistic data*, which has been firstly proposed in [6]. Basically, [6] introduces the *possible-world semantics* concept, according to which, given a probabilistic database D_P, D_P can be represented and processed (e.g., during query evaluation) via admitting the existence of *different possible-world databases* $D_{P,k}$, each one obtained from D_P via assigning *possible values* to probabilistic attributes in D_P among those modeled by the respective confidence intervals, based on the associated probabilities. It would be clear enough to notice that this approach curs the risk of generating an *exponential* number of possible-world databases (i.e., $k \to e^{|D_P|}$, such that $|D_P|$ denotes the cardinality of D_P), even because a *combinatory dependence* among probabilistic attributes of D_P exists. Based on this model for representing probabilistic databases, [6] finally retrieves the output data cube over D_P, denoted by $C(D_P)$, via *estimating* aggregates (to be stored within data cube cells of $C(D_P)$) from the universe of possible-world databases derived from D_P by means of probabilistic/statistical *estimation tools and techniques* [31] and via detecting several *probability-inspired consistency conditions* [6].

While there is a wide and rich literature on the issue of *efficiently processing and querying probabilistic databases* (see Sect. 3), despite the relevance of OLAP applications for next-generation *Data Warehousing* (DW) *and Business Intelligence* (BI) *systems* very few papers address at now the yet-interesting problem of computing and querying OLAP data cubes over probabilistic data (e.g., [6,7,8]). Contrary to this actual trend, it is natural to foresee that this problem will play more and more a leading role in the context of DW and BI systems, due to the obvious popularity of uncertain and imprecise datasets (e.g., environmental sensor networks, data stream management systems, alarm and surveillance systems, RFID-based applications, supply-chain management systems).

Inspired by these motivations, starting from limitations of [6] in this paper we propose a framework for efficiently computing and querying multidimensional OLAP data cubes over probabilistic data, which we name as *probabilistic data cubes*. Most distinctive contributions of our research are the following: (*i*) a meaningful *decomposition approach* that allows us to extract the so-called *decomposed probabilistic database* from the input probabilistic database at the cost of a *sub-linear complexity* – the decomposed probabilistic database is then used to compute the final probabilistic data cube based on conventional *multidimensional aggregation methods* [20]; (*ii*) a novel *Probability Distribution Function* (PDF) [31]-based model, inspired by [36], that allows us to query probabilistic data cubes efficiently.

2 Problem Formulation

Given a probabilistic database D_P modeled in terms of a collection of probabilistic relations R_i, i.e. $D_P = \{R_0, R_1, \ldots, R_{|D_P|-1}\}$, the problem we investigate in this research consists in effectively and efficiently computing and querying a data cube over D_P, $C(D_P)$, given an input *data cube schema* [37] W. According to the nature of D_P, we properly define $C(D_P)$ as a probabilistic data cube (we detail this novel definition next). Now, focus the attention on the class of probabilistic databases considered in our research, which is inspired from fundamental works in [4,2,34]. Given a probabilistic relation R_i in D_P modeled in terms of a collection of attributes, i.e. $R_i = \{A_{i,0}, A_{i,1}, \ldots, A_{i,|R_i|-1}\}$, such that A_{i,k_j}, with k_j in $\{0, 1, \ldots, |R_i| - 1\}$, denotes an attribute in R_i, two distinct sub-set of attributes in R_i can be identified. The first one, denoted by $R_i^E \subset R_i$, such that $R_i^E = \{A_{i,k_0}^E, A_{i,k_1}^E, \ldots, A_{i,k_{|R_i^E|-1}}^E\}$, with k_j in $\{0, 1, \ldots, |R_i|$ $- 1\}$, stores the sub-set of *exact attributes* in R_i, i.e. attributes in R_i whose values are exact. The second one, denoted by $R_i^P \subset R_i$, such that $R_i^P = \{A_{i,k_0}^P, A_{i,k_1}^P, \ldots, A_{i,k_{|R_i^P|-1}}^P\}$, with k_j in $\{0, 1, \ldots, |R_i| - 1\}$ stores the sub-set of probabilistic attributes in R_i, i.e. attributes in R_i whose values are probabilistic. Obviously, $R_i^E \cap R_i^P = \emptyset$. An exact attribute A_{i,k_j}^E in R_i^E is defined as follows: $A_{i,k_j}^E = \left\{V_{i,k_j}^E \middle| V_{i,k_j}^E \in \mathbb{D}_{i,k_j}^E\right\}$, where V_{i,k_j}^E denotes an exact value of A_{i,k_j}^E and $\mathbb{D}_{i,k_j}^E$ the domain of A_{i,k_j}^E, respectively. A probabilistic attribute A_{i,k_j}^P in R_i^P is defined as follows:

$$A_{i,k_j}^P = \left\{\langle [V_{i,k_{j,min}}^P, V_{i,k_{j,max}}^P], p_{i,k_j}\rangle \middle| V_{i,k_{j,min}}^P \in \mathbb{D}_{i,k_j}^P, V_{i,k_{j,max}}^P \in \mathbb{D}_{i,k_j}^P, \right.$$
$$\left. V_{i,k_{j,min}}^P < V_{i,k_{j,max}}^P, 0 \le p_{i,k_j} \le 1\right\} \tag{1}$$

such that: (*i*) $V_{i,k_{j,min}}^P$ and $V_{i,k_{j,max}}^P$ denote probabilistic values of A_{i,k_j}^P, respectively; (*ii*) $[V_{i,k_{j,min}}^P, V_{i,k_{j,max}}^P]$ denotes the confidence interval associated to A_{i,k_j}^P; (*iii*) p_{i,k_j} denotes the probability associated to $[V_{i,k_{j,min}}^P, V_{i,k_{j,max}}^P]$; (*iv*) $\mathbb{D}_{i,k_j}^P$ denotes the domain of A_{i,k_j}^P. The semantics of this confidence-interval-based model states that the possible value of A_{i,k_j}^P ranges between $V_{i,k_{j,min}}^P$ and $V_{i,k_{j,max}}^P$ with probability p_{i,k_j}. Also, a law describing the *probability distribution* according to which possible values of A_{i,k_j}^P vary over the interval $[V_{i,k_{j,min}}^P, V_{i,k_{j,max}}^P]$ is assumed. Without loss of generality, the *Uniform distribution* [12] is very often taken as reference. The Uniform distribution states that (possible) values in $[V_{i,k_{j,min}}^P, V_{i,k_{j,max}}^P]$ have *all* the same probability of being the exact value of A_{i,k_j}^P, denoted by V_{i,k_j}, actually, i.e.:

$$P\left(A_{i,k_j}^P = V_{i,k_j}\right) = P\left(A_{i,k_{j+1}}^P = V_{i,k_j}\right) = \frac{p_{i.k_j}}{\left|V_{i,k_{j,max}}^P - V_{i,k_{j,min}}^P\right|}$$
$$\forall\, k_j, k_{j+1}: k_j \neq k_{j+1}, k_j \in \{0,1,\ldots,|R_i^P| - 1\}, k_{j+1} \in \{0,1,\ldots,|R_i^P| - 1\} \tag{2}$$

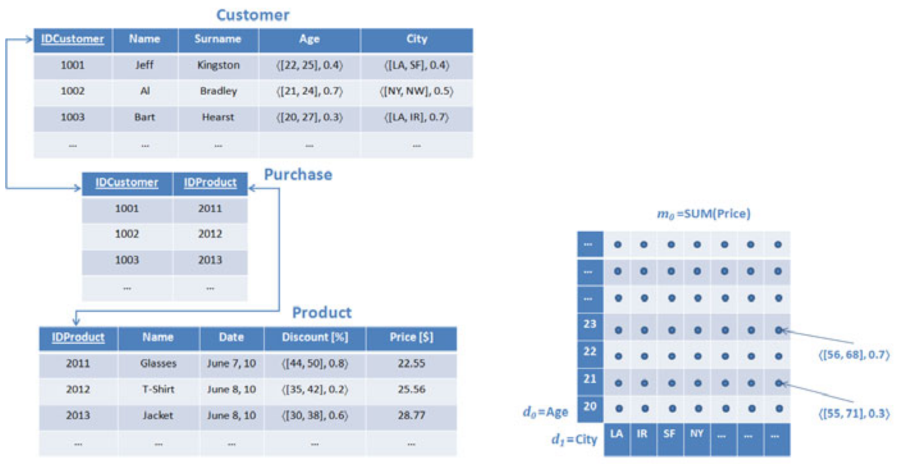
Despite the popularity of the Uniform distribution, the confidence-interval-based model above is prone to incorporate any other kind of state-of-the-art probability distribution [31].

Given a data cube C, the data cube schema of C, W, is defined as the tuple: $W = \langle D, H, M \rangle$, such that [20]: (*i*) $D = \{d_0, d_1, \ldots, d_{|D|-1}\}$ denotes the set of dimensions of C; (*ii*) $H = \{h_0, h_1, \ldots, h_{|H|-1}\}$ denotes the set of hierarchies of C, being h_i in H the hierarchy associated to the dimension d_i in D; (*iii*) M denotes the set of measures of C. As directly related to data cubes measures, a SQL aggregation operator (e.g., SUM, COUNT, AVG) is set as the baseline operation of the aggregation process generating data cube cells [20]. Without loss of generality, in this paper we consider as reference the class of SUM-based data cubes, which is general enough to capture a wide spectrum of real-life DW and BI applications [15]. In addition to this, as regards the specific in-memory data representation, we refer to *MOLAP data cubes* [21], i.e. data cubes represented in terms of *multidimensional arrays*, a general format to which any alternative in-memory data cube representation technique (e.g., ROLAP, HOLAP) may converge easily [21]. Also, for the sake of simplicity, we hereby assume of dealing with single-measure data cubes, i.e. $M = \{m_0\}$, and discard the case of *multiple-measure data cubes* [18], i.e. M such that $|M| > 1$, as the latter case can be straightforwardly obtained via extending actual models and algorithms provided for single-measure data cubes. Dimensions $d_0, d_1, \ldots, d_{|D|-1}$ in D and the measure m_0 in M are defined from attributes in D_P, meaning that the DW administrator assigns the role of dimension to a partition of attributes in D_P whereas he/she assigns the role of measure to one attribute in D_P. This is what is commonly intended as a *multidimensional abstraction of relational data* [20]. For the sake of simplicity, in our research we assume that the measure m_0 is chosen among exact attributes in D_P, i.e. $m_0 \in R_i^E \subset R_i$, such that $R_i \in D_P$. Contrary to the constraint on the measure m_0, dimensions in D can instead be chosen among both exact and probabilistic attributes in D_P, i.e. $d_i \in R_i^E \subset R_i$ or $d_i \in R_i^P \subset R_i$, such that $R_i \in D_P$.

Given a probabilistic data cube C, each cell of C, denoted by $c(i_0, i_1, \ldots, i_{|D|-1}) = C[i_0][i_1]\ldots[i_{|D|-1}]$, such that $i_0 \in \{0, 1, \ldots, |d_0| - 1\}$, $i_1 \in \{0, 1, \ldots, |d_1| - 1\}$, …, $i_{|D|-1} \in \{0, 1, \ldots, |d_{|D|-1}| - 1\}$, is probabilistic in nature, according to D_P. Formally, $c(i_0, i_1, \ldots, i_{|D|-1}) = \langle [B_{min}^{c(i_0..i_{|D|-1})}, B_{max}^{c(i_0..i_{|D|-1})}], p_{c(i_0..i_{|D|-1})} \rangle$, such that: (*i*) $[B_{min}^{c(i_0..i_{|D|-1})}, B_{max}^{c(i_0..i_{|D|-1})}]$ denotes the confidence interval (over *aggregate values*) associated to $c(i_0, i_1, \ldots, i_{|D|-1})$, with $B_{min}^{c(i_0..i_{|D|-1})} < B_{max}^{c(i_0..i_{|D|-1})}$; (*ii*) $p_{c(i_0..i_{|D|-1})}$ denotes the probability associated to $[B_{min}^{c(i_0..i_{|D|-1})}, B_{max}^{c(i_0..i_{|D|-1})}]$, with $0 \leq p_{c(i_0..i_{|D|-1})} \leq 1$. As a first result, it should be noted that our notion of probabilistic data cube is novel under the classical notion proposed in [6], which aims at outputting exact data cubes.

To give a practical example, consider Fig. 1, where a probabilistic database storing purchase data (Fig. 1 (*a*)) and a two-dimensional probabilistic data cube (Fig. 1 (*b*)) built on top of such a database are depicted, respectively. In the database of Fig. 1 (*a*), (probabilistic) table *Customer* stores information on customers making purchases. In particular, *Customer* is characterized by two probabilistic attributes, namely *Age*, which models the age of customers, and *City*, which models the city of customers.

(Probabilistic) table *Product* stores information on the available products. In particular, *Product* is characterized by a singleton probabilistic attribute, namely *Discount*, which models the discount applied to products. Finally, (exact) table *Purchase* represents associations among tuples stored in *Customer* and tuples stored in *Product*, respectively. In the data cube of Fig. 1 (*b*), (probabilistic) attributes *Age* and *City* play the role of dimensions d_0 and d_1, respectively, whereas (exact) attribute *Price* plays the role of measure m_0, being SUM the basic SQL aggregation operator. In particular, Fig. 1 (*b*) shows the measure values (i.e., aggregate values) of two distinct (probabilistic) data cube cells, with associated confidence intervals and probabilities, respectively.



Customer

IDCustomer	Name	Surname	Age	City
1001	Jeff	Kingston	⟨[22, 25], 0.4⟩	⟨[LA, SF], 0.4⟩
1002	Al	Bradley	⟨[21, 24], 0.7⟩	⟨[NY, NW], 0.5⟩
1003	Bart	Hearst	⟨[20, 27], 0.3⟩	⟨[LA, IR], 0.7⟩
...	...	...	...	...

Purchase

IDCustomer	IDProduct
1001	2011
1002	2012
1003	2013
...	...

Product

IDProduct	Name	Date	Discount [%]	Price [$]
2011	Glasses	June 7, 10	⟨[44, 50], 0.8⟩	22.55
2012	T-Shirt	June 8, 10	⟨[35, 42], 0.2⟩	25.56
2013	Jacket	June 8, 10	⟨[30, 38], 0.6⟩	28.77
...	...	...	...	...

(a)

(b)

Fig. 1. An example probabilistic database storing purchase data (*a*) and a two-dimensional probabilistic data cube (*b*) built on top of such a database

As highlighted in Sect. 1, beyond to the problem of computing probabilistic data cubes from probabilistic databases, in our research we also investigate the issue of querying probabilistic data cubes efficiently. First, we fix the class of OLAP queries of interest for our research. Among several alternatives, *range aggregation queries* [24] represent a popular and useful solution for extracting summarized knowledge from OLAP data cubes [15]. Given a $|D|$-dimensional OLAP data cube C and a target SQL aggregation operator F (e.g., SUM, COUNT, AVG), an M-dimensional range-F query Q over C is defined as the tuple: $Q = \langle L, F\rangle$, such that: (*i*) L denotes a multidimensional range over C, i.e. $L = \langle [l_{k_0}, u_{k_0}], [l_{k_1}, u_{k_1}], \ldots, [l_{k_{M-1}}, u_{k_{M-1}}]\rangle$, with k_i in $\{0, 1, \ldots, |D| - 1\}$, where l_{k_i} and u_{k_i}, with $l_{k_i} < u_{k_i}$, model a lower bound and an upper bound over the dimension d_{k_i} in D, respectively; (*ii*) F denotes the target SQL aggregation operator. Evaluated against C, Q applies F over the domain of *all* the data cubes cells in C bounded by L, denoted by $C(\dashv L)$ and defined as follows: $C(\dashv L) = \{c(i_{k_0}, i_{k_1}, \ldots, i_{k_{M-1}}) \in C \mid \{c(i_{k_0}, i_{k_1}, \ldots, i_{k_{M-1}}) \subset L,\ l_{k_0} \leq i_{k_0} \leq$

u_{k_0}, $l_{k_1} \leq i_{k_1} \leq u_{k_1}, \ldots, l_{k_{M-1}} \leq i_{k_{M-1}} \leq u_{k_{M-1}}\}$, and retrieves a *probabilistic aggregate value*. Similarly to the probabilistic model for representing data cube cells of (probabilistic) data cubes above, in our framework we model the *probabilistic answer* to an OLAP query Q, denoted by $A^P(Q)$, over a probabilistic data cube C as the tuple: $A^P(Q) = \langle [U^P_{Q,min}, U^P_{Q,max}], p_Q \rangle$, such that: (*i*) $[U^P_{Q,min}, U^P_{Q,max}]$ denotes the confidence interval associated to Q, with $U^P_{Q,min} < U^P_{Q,max}$; (*ii*) p_Q denotes the probability associated to $[U^P_{Q,min}, U^P_{Q,max}]$, with $0 \leq p_Q \leq 1$. To give an example, consider Fig. 2, where two range-SUM queries (and related probabilistic answers) Q_i and Q_j over the two-dimensional probabilistic data cube of Fig. 1 (*b*) are depicted, respectively.

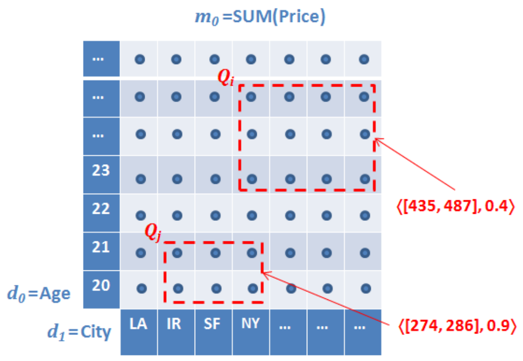


Fig. 2. Two example range-SUM queries over the two-dimensional probabilistic data cube of Fig. 1 (*b*)

As we demonstrate in Sect. 5, a relevant contribution of our research is represented by the innovative approach for evaluating range queries over probabilistic data cubes in order to retrieve probabilistic answers.

3 Related Work

The problem of effectively and efficiently processing and querying probabilistic data has recently gained a great deal of attention from the database research community, due to the wide spread of application scenarios where probabilistic data occur. It would be clear enough how the latter issue plays a fundamental role for our main research context (i.e., efficiently computing and querying OLAP data cubes over probabilistic data), hence in this Section we initially focus the attention on probabilistic database research issues. Then, we move the attention on state-of-the-art proposals directly related to our research.

A first problem investigated by authoritative research initiatives deals with the issue of *representing probabilistic data*, which mainly reflects the intrinsic nature of such kind of data sources characterized by uncertainty and imprecision. In this research setting, [3] puts theoretical and conceptual bases for the management of probabilistic data, hence constituting a milestone for sub-sequent research efforts, such as [17] that further extends foundations proposed in [3] by also defining new unexplored challenges, mainly from a theoretical point of view. Basically, two main classes of proposals for representing probabilistic databases exist. First one introduces the so-called *probabilistic database model* [4,2,34], which essentially embeds probabilities within relational tables directly, like in our reference model (see Sect. 2) that is in fact inspired from these research initiatives. Second class of proposals for representing probabilistic databases instead pursues the so-called *uncertain object model* [10,16,17,32], according to which uncertain objects over probabilistic databases that can be modeled in terms of *sample-based* PDF [31] are introduced and exploited for data representation and query evaluation purposes.

Querying probabilistic data is obviously one of the hot topics in probabilistic database research, due to the fact different kinds of *query semantics* can be defined on top of such databases, and the query evaluation process itself can be "customized" accordingly. Different query semantics introduce indeed different benefits and limitations, but several aspects remain unsolved, so that this research issue can be still considered as an open problem. In this research setting, [10], on the basis of the notion of uncertain objects, defines suitable query strategies for probabilistic databases that directly exploit statistical properties of model PDF. Then, [16] completes previous results via finding *safety conditions* for SQL plans derived from queries over probabilistic databases, and [32] studies *approximate query evaluation* opportunities on top of such databases. Based on the above-mentioned experiences, database researchers have devoted a significant deal of attention to the problem of effectively and efficiently evaluating a wide family of queries that can be posed over probabilistic databases, among which noticeable ones are the following: (*i*) *join queries* (e.g., [11,28]); (*ii*) *top-k queries* (e.g., [35,38]); (*iii*) *rank queries* (e.g., [29,25]).

For what regards the main problem of efficiently computing and querying OLAP data cubes over probabilistic data, we can identify two classes of state-of-the-art proposals directly related to our research. First one comprises a collection of contributions that deal with the issue of *efficiently evaluating aggregates over probabilistic databases*. This line of research is relevant for our context as computing aggregates over probabilistic data can be indented as the baseline operation for computing (probabilistic) aggregate values to be stored within (probabilistic) data cube cells (see Sect. 2). [9] first puts formal bases of the issue of supporting efficient aggregate query evaluation over imprecise data, a significant sub-set of probabilistic data (see Sect. 1). [30] moves the attention on the most relevant case of computing aggregates over uncertain and imprecise databases, and proposes a novel *model-based approach* to this end. Finally, [33] investigates theoretical and complexity aspects of computing aggregates over probabilistic databases. Second class of proposals directly related to our research match perfectly the same problem investigated by us, i.e. computing and querying OLAP data cubes over probabilistic data. As highlighted in Sect. 1, [6] first introduces the problem of effectively and efficiently computing OLAP data cubes over probabilistic data, hence it has to be considered as the state-of-the-art proposal in our reference research context. According to [6], the ambiguity of uncertain and imprecise tuples of a given probabilistic database D_P can be solved by admitting the existence of a family of different possible worlds $D_{P,k}$ built on top of D_P, such that, in each possible world $D_{P,k}$, probabilistic tuples *alternatively* take (exact) values of probabilistic attributes among those modeled by the associated confidence intervals. Also, a *weight* w_k is associated to each possible world $D_{P,k}$, in order to capture the *likelihood* of $D_{P,k}$ of actually being the "true" world among the possible ones [6]. As stated in [6], the number of possible worlds can become exponential, also due to the combinatory dependence among probabilistic attributes of D_P. In more detail, given a probabilistic database D_P and a relation R_i in D_P having $|R_i^P|$ probabilistic attributes $R_i^P = \{A_{i,k_0}^P, A_{i,k_1}^P, \ldots, A_{i,k_{|R_i^P|-1}}^P\}$, with k_j in $\{0, 1, \ldots, |R_i| - 1\}$, R_i leads to $\prod_{j=0}^{|R_i^P|-1} \left|\mathbb{D}_{i,k_j}^P\right|$ possible worlds that can be extracted from D_P [6], being $\mathbb{D}_{i,k_j}^P$ the domain of A_{i,k_j}^P. For a probabilistic database D_P having P probabilistic relations, such that $P \leq |D_P|$, the number of possible worlds is: $\prod_{i=0}^{P-1} \prod_{j=0}^{|R_i^P|-1} \left|\mathbb{D}_{i,k_j}^P\right|$, which is clearly not acceptable for real-life database environments.

A critical aspect discussed in [6] is represented by the so-called *allocation policies*, which determine how to compute the probability to be assigned to uncertain and imprecise tuples. In more detail, this probability captures the likelihood of a given uncertain and imprecise tuple of being aggregated within one of the measures of the target OLAP cube. In this respect, several alternatives are proposed in [6] and, sub-sequentially, extended by the same authors in [7]. Finally, in [8] authors address the problem of efficiently computing and querying OLAP data cubes over probabilistic data in the presence of *constraints* on attribute domains by further exploiting and extending the baseline solution for free-of-constraint attribute domains given in [6].

4 Computing Probabilistic OLAP Data Cubes

In this Section, we provide models and algorithms for computing probabilistic OLAP data cubes from probabilistic databases, as one of the most prominent contribution of our research. Given a probabilistic database $D_P = \{R_0, R_1, \ldots, R_{|D_P|-1}\}$, and a data cube schema $W = \langle D, H, M \rangle$, the probabilistic data cube $C(D_P)$ over D_P is defined by means of a *multidimensional mapping scheme* $\mathbb{M}_{D_P \to C(D_P)} = \langle \mathbb{M}_D, \mathbb{M}_H, \mathbb{M}_M \rangle$, such that: (*i*) $\mathbb{M}_D : \{R_0, \ldots, R_{|D_P|-1}\} \to \{d_0, \ldots, d_{|D|-1}\}$ represents a mapping (sub-)scheme between attributes in D_P and dimensions in $D \in W$, such that $A_{i,k_j} \to d_{h_l}$ denotes a mapping between an attribute A_{i,k_j} of a relation R_i in D_P and a dimension d_{h_l} in D, with k_j in $\{0, 1, \ldots, |R_i| - 1\}$ and h_l in $\{0, 1, \ldots, |D| - 1\}$; (*ii*) $\mathbb{M}_H : \{R_0, \ldots, R_{|D_P|-1}\} \to \{h_0, \ldots, h_{|H|-1}\}$ represents a mapping (sub-)scheme between attributes in D_P and hierarchies in $H \in W$, such that $A_{i,k_j} \to h_{v_m}$ denotes a mapping between an attribute A_{i,k_j} of a relation R_i in D_P and a hierarchy h_{v_m} in H, with k_j in $\{0, 1, \ldots, |R_i| - 1\}$ and v_m in $\{0, 1, \ldots, |H| - 1\}$; (*iii*) $\mathbb{M}_M : \{R_0, \ldots, R_{|D_P|-1}\} \to \{m_0\}$ represents a mapping (sub-)scheme between an attribute A_{i,k_j} of a relation R_i in D_P and the measure m_0 in $M \in W$ (from Sect. 2, recall that $|M| = 1$), denoted by $A_{i,k_j} \to m_0$, with k_j in $\{0, 1, \ldots, |R_i| - 1\}$. For the sake of simplicity, in this research we investigate the case of computing probabilistic data cubes such that $\mathbb{M}_H = \emptyset$, which determines the special case of data cubes without hierarchies. It should be noted that such data cubes are general enough to capture a wide family of real-life applications, as clearly highlighted in [14]. On the other hand, models and algorithms presented in our paper can be easily extended as to deal with more significant hierarchy-equipped data cubes.

Before presenting our proposed approach for computing probabilistic data cubes, we need to recall a well-known concept in *Data Mining* (DM), i.e. *discretization of relational database attributes* (e.g., [21]). Given a database D and an attribute A_{i,k_j} of a relation R_i in D, such that k_j in $\{0, 1, \ldots, |R_i| - 1\}$, having $\mathbb{D}_{i,k_j}$ as domain, the *discretized domain* of A_{i,k_j}, denoted by $\mathcal{D}(A_{i,k_j}) = \{\rho_{i,k_j,0}, \rho_{i,k_j,1}, \ldots, \rho_{i,k_j,|\mathcal{D}(A_{i,k_j})|-1}\}$, is defined as a collection of *discretized members* $\rho_{i,k_j,h}$ derived from $\mathbb{D}_{i,k_j}$ on the basis of the nature of A_{i,k_j}. If A_{i,k_j} is numeric in nature, then $\mathcal{D}(A_{i,k_j})$ is composed by all the numeric members extracted from the domain that is in a *bijective relation* with $\mathbb{D}_{i,k_j}$ starting from the minimum value in A_{i,k_j}, denoted by V_{i,k_j}^{MIN}, to the maximum value in A_{i,k_j}, denoted by V_{i,k_j}^{MAX}. For instance, if A_{i,k_j} stores positive integer values, then $\mathbb{D}_{i,k_j}$ is in a bijective relation with the domain of natural numbers $\mathbb{N}$ and $\mathcal{D}\left(A_{i,k_j}\right) = \left\{V_{i,k_j}^{MIN}, V_{i,k_j}^{MIN} + 1, \ldots, V_{i,k_j}^{MAX} - 1, V_{i,k_j}^{MAX}\right\}$. If A_{i,k_j} is categorical in nature, then a *topological ordering relation* $\prec$ over categorical members in $\mathbb{D}_{i,k_j}$ must be introduced. Note that, in the previous example, $\prec \equiv <$. On the basis of the ordering determined by $\prec$, $\mathcal{D}(A_{i,k_j})$ is composed by all the categorical members in $\mathbb{D}_{i,k_j}$ starting from the "minimum" member in A_{i,k_j}, V_{i,k_j}^{MIN}, to the "maximum" member in A_{i,k_j}, V_{i,k_j}^{MAX}. For instance, if A_{i,k_j} stores the week days, then $\mathcal{D}\left(A_{i,k_j}\right) = \{Sunday, Monday, \ldots, Saturday\}$ (e.g., $Sunday \prec Monday$).

On the basis of the formal DM framework above, given a database D and an attribute A_{i,k_j} of a relation R_i in D, such that k_j in $\{0, 1, \ldots, |R_i| - 1\}$, having $\mathbb{D}_{i,k_j}$ as domain and $\prec$ as topological ordering relation, we introduce the function nextM that takes as input A_{i,k_j}, $\mathbb{D}_{i,k_j}$, $\prec$ and a member V_{i,k_j} in $\mathcal{D}(A_{i,k_j})$, and returns as output the member V_{i,k_j}^{SUC} that "follows" V_{i,k_j} in $\mathcal{D}(A_{i,k_j})$ based on the ordering determined by $\prec$. Formally, nextM(A_{i,k_j}, $\mathbb{D}_{i,k_j}$,$\prec$, V_{i,k_j}) = V_{i,k_j}^{SUC}, such that $V_{i,k_j} \prec V_{i,k_j}^{SUC}$.

Now, focus the attention on how the probabilistic data cube $C(D_P)$ is finally computed from the input probabilistic database D_P and data cube schema W. Let $D = \{d_0, \ldots, d_{|D|-1}\} = \left\{A_{i,k_0}^E, A_{i,k_1}^E, \ldots, A_{i,k_{|E|-1}}^E, A_{i,k_{|E|}}^P, A_{i,k_{|E|+1}}^P, \ldots, A_{i,k_{|P|-|E|-1}}^P\right\}$ be the set of $|D| = |P| - |E|$ dimensions in W, selected from $|E|$ exact attributes and $|P|$ probabilistic attributes in D_P (see Sect. 2), respectively, such that (*i*) A_{i,k_j}^E, with k_j in $\{0, 1, \ldots, |R_i| - 1\}$, being R_i a relation in D_P, denotes an exact attribute in D_P and (*ii*) A_{i,k_j}^P, with k_j in $\{0, 1, \ldots, |R_i| - 1\}$, being R_i a relation in D_P, denotes a probabilistic attribute in D_P. Let D^E denote the set of exact attributes/dimensions in D and D^P the set of probabilistic attributes/dimensions in D, respectively ($D^E \cap D^P = \emptyset$). Let $M = \{m_0\} = \{\hat{A}_{i,k_j}^E\}$ be the singleton measure in W, such that $\hat{A}_{i,k_j}^E$, with k_j in $\{0, 1, \ldots, |R_i| - 1\}$, being R_i a relation in D_P, denotes an exact attribute in D_P (from Sect. 2, recall that, in our probabilistic data cube model, measures are always chosen among exact attributes in D_P).

The approach for computing $C(D_P)$ we propose is two-step in nature. In the first step, the so-called decomposed probabilistic database, denoted by $D_P^{\triangleright}$, is obtained from D_P directly by means of an innovative *decomposition technique* whose main aim consists in *breaking confidence intervals of probabilistic attribute values, while preserving the probabilistic nature of tuples in D_P.* In the second step, $C(D_P)$ is finally computed from $D_P^{\triangleright}$ based on conventional multidimensional aggregation methods [20].

First, we describe the proposed approach for extracting $D_P^{\triangleright}$ from D_P. Basically, for each *dimensional attribute* A_{i,k_j} of relations R_i in D_P, in the first step we decompose attribute values V_{i,k_j} of A_{i,k_j} into a set of *decomposed attribute values*, denoted by $\mathcal{P}(A_{i,k_j})$, depending on the nature of A_{i,k_j} (exact, or probabilistic) and by exploiting the *multidimensional relation* with the corresponding attribute value $\hat{V}_{i,k_j}$ of the *measure attribute* $\hat{A}_{i,k_j}$ of relations R_i in D_P. Then, decomposed attribute value sets $\mathcal{P}(A_{i,k_j})$ are used to populate $D_P^{\triangleright}$ and, finally, $C(D_P)$ is aggregated from $D_P^{\triangleright}$ directly, based on the input data cube schema W. It should be noted that, depending on W, since OLAP dimensions and measures are typically a *sub-set* of the whole attribute set of the input database [20], without loss of generality we observe that $|D_P^{\triangleright}| < |D_P|$. This nice amenity contributes to lower the *spatial complexity* of the approach we propose.

Let: (*i*) $d_i \equiv A_{i,k_j}^E$ be an exact dimension in W; (*ii*) $m_0 \equiv \hat{A}_{i,k_j}^E$ be the exact measure in W; (*iii*) $A_{i,k_j}^E[\ell] = V_{i,k_j}^E$ be an exact value of A_{i,k_j}^E at position ℓ, such that $0 \leq \ell \leq \left|\mathbb{D}_{i,k_j}^E\right| - 1$, being $\mathbb{D}_{i,k_j}^E$ the domain of A_{i,k_j}^E; (*iv*) $\hat{A}_{i,k_j}^E[\ell] = \hat{V}_{i,k_j}^E$ be the corresponding exact value of $\hat{A}_{i,k_j}^E$ at position ℓ, such that $0 \leq \ell \leq \left|\hat{\mathbb{D}}_{i,k_j}^E\right| - 1$, being $\hat{\mathbb{D}}_{i,k_j}^E$ the domain of $\hat{A}_{i,k_j}^E$. Since d_i is exact, decomposing the pair $\langle d_i, m_0\rangle \equiv \langle A_{i,k_j}^E, \hat{A}_{i,k_j}^E\rangle$ generates the following set of decomposed attribute values $\mathcal{P}\left(A_{i,k_j}^E\right) = \left\{\langle V_{i,k_j}^E, 1, \hat{V}_{i,k_j}^E\rangle\right\}$. Note that $\left|\mathcal{P}\left(A_{i,k_j}^E\right)\right| = 1$. Contrary to the latter case, let: (*i*) $d_i \equiv A_{i,k_j}^P$ be a probabilistic dimension in W; (*ii*) $m_0 \equiv \hat{A}_{i,k_j}^E$ be the exact measure in W; (*iii*) $A_{i,k_j}^P[\ell] = \langle\left[V_{i,k_{j,min}}^P, V_{i,k_{j,max}}^P\right], p_{i,k_j}\rangle$ be a probabilistic value of A_{i,k_j}^P at position ℓ, such that $0 \leq \ell \leq \left|\mathbb{D}_{i,k_j}^P\right| - 1$, being $\mathbb{D}_{i,k_j}^E$ the domain of A_{i,k_j}^P; (*iv*) $\hat{A}_{i,k_j}^E[\ell] = \hat{V}_{i,k_j}^E$ be the corresponding exact value of $\hat{A}_{i,k_j}^E$ at position ℓ, such that $0 \leq \ell \leq \left|\hat{\mathbb{D}}_{i,k_j}^E\right| - 1$, being $\hat{\mathbb{D}}_{i,k_j}^E$ the domain of $\hat{A}_{i,k_j}^E$. Since d_i is probabilistic, decomposing the pair $\langle d_i, m_0\rangle \equiv \langle A_{i,k_j}^P, \hat{A}_{i,k_j}^E\rangle$ generates the following set of decomposed attribute values:

$$\mathcal{P}\left(A_{i,k_j}^{P}\right) = \left\{ \begin{array}{l} \langle V_{i,k_{j,min}}^{P}, \frac{p_{i,k_j}}{\left|V_{i,k_{j,max}}^{P} - V_{i,k_{j,min}}^{P}\right|}, \frac{\hat{V}_{i,k_j}^{E}}{\left|V_{i,k_{j,max}}^{P} - V_{i,k_{j,min}}^{P}\right|} \rangle, \\ \langle \mathrm{nextM}\left(A_{i,k_j}^{P}, \mathbb{D}_{i,k_j}^{E}, \prec, V_{i,k_{j,min}}^{P}\right), \frac{p_{i,k_j}}{\left|V_{i,k_{j,max}}^{P} - V_{i,k_{j,min}}^{P}\right|}, \\ \frac{\hat{V}_{i,k_j}^{E}}{\left|V_{i,k_{j,max}}^{P} - V_{i,k_{j,min}}^{P}\right|} \rangle, \\ \ldots, \\ \langle V_{i,k_{j,max}}^{P}, \frac{p_{i,k_j}}{\left|V_{i,k_{j,max}}^{P} - V_{i,k_{j,min}}^{P}\right|}, \frac{\hat{V}_{i,k_j}^{E}}{\left|V_{i,k_{j,max}}^{P} - V_{i,k_{j,min}}^{P}\right|} \rangle \end{array} \right\} \tag{3}$$

Note that $\left|\mathcal{P}\left(A_{i,k_j}^{P}\right)\right| = \left|\mathcal{D}\left(A_{i,k_j}^{P}\right)_{\left[V_{i,k_{j,min}}^{P}:V_{i,k_{j,max}}^{P}\right]}\right|$, such that $I_{[I_{min}:I_{max}]}$ denotes the *sub-set operator* that, given a set I, extracts from I the sub-set spanning from $I_{min} \in I$ to $I_{max} \in I$.

For each dimensional attribute value in W, the decomposition task implemented by the first step of the approach we propose generates in $D_P^{\rhd}$ a decomposed attribute value set storing tuples of kind: $\langle V_{i,k_j}^{\rhd}, p_{i,k_j}^{\rhd}, \hat{V}_{i,k_j}^{\rhd} \rangle$. By composing such tuples for *all* the $|D|$ dimensional attributes in W, we finally obtain the generic tuple stored in $D_P^{\rhd}$ as follows: $\langle \langle V_{i,k_0}^{\rhd}, p_{i,k_0}^{\rhd}, \hat{V}_{i,k_0}^{\rhd} \rangle, \langle V_{i,k_1}^{\rhd}, p_{i,k_1}^{\rhd}, \hat{V}_{i,k_1}^{\rhd} \rangle, \ldots, \langle V_{i,k_{|D|-1}}^{\rhd}, p_{i,k_{|D|-1}}^{\rhd}, \hat{V}_{i,k_{|D|-1}}^{\rhd} \rangle \rangle$. It should be noted that, with respect to the original database schema of D_P and the input data cube schema W, the (database) schema of $D_P^{\rhd}$ only maintains the dimensional attributes d_0, d_1, …, $d_{|D|-1}$ and discards the (singleton) measure attribute m_0. This because the contribution of measure attribute values in D_P is “distributed” across decomposed attribute values in $D_P^{\rhd}$. Again, similarly to the previous consideration on the cardinality of $D_P^{\rhd}$ with respect to the cardinality of D_P, it should be clear enough that this further nice amenity contributes to lower the final spatial complexity of $D_P^{\rhd}$.

In the second step of our proposed approach for computing probabilistic OLAP data cubes, the final data cube $C(D_P)$ is aggregated from $D_P^{\rhd}$ directly based on conventional multidimensional aggregation methods [20] plus the novelty represented by an *innovative technique for computing confidence intervals and probabilities of probabilistic data cube cells* (see Sect. 2). For the sake of simplicity, let us to denote as $D = \{d_0, \ldots, d_{|D|-1}\} = \left\{A_{i,k_0}, A_{i,k_1}, \ldots, A_{i,k_{|D|-1}}\right\}$ the set of $|D|$ dimensions in W, by removing the notation allowing us to distinguish between exact and probabilistic dimensional attributes, due the fact that, as shown above, the decomposition task implemented by the first step of our proposed approach treats both kinds of attributes in a unified manner. Equally, we could refer to the schema of $D_P^{\rhd}$ as: $D_P^{\rhd}\left(A_{i,k_0}, A_{i,k_1}, \ldots, A_{i,k_{|D|-1}}\right)$. Let $\mathcal{D}(A_{i,k_0}), \mathcal{D}(A_{i,k_1}), \ldots, \mathcal{D}(A_{i,k_{|D|-1}})$ be the discretized domains of $A_{i,k_0}, A_{i,k_1}, \ldots, A_{i,k_{|D|-1}}$, respectively. Based on conventional multidimensional aggregation methods [20], for each multidimensional entry $\boldsymbol{h_l} = \left(\rho_{i,k_0,h_0}, \rho_{i,k_1,h_1}, \dots, \rho_{i,k_{|D|-1},h_{|D|-1}}\right)$ in the *multidimensional space* defined by the *Cartesian product* $\mathcal{D}(A_{i,k_0}) \times \mathcal{D}(A_{i,k_1}) \times \dots \times \mathcal{D}(A_{i,k_{|D|-1}})$, such that $h_0 \in \{0, 1, \dots, |\mathcal{D}(A_{i,k_0})| - 1\}$, $h_1 \in \{0, 1, \dots, |\mathcal{D}(A_{i,k_1})| - 1\}$, …, $h_{|D|-1} \in \{0, 1, \dots, |\mathcal{D}(A_{i,k_{|D|-1}})| - 1\}$, a set of attribute values $\Psi(\boldsymbol{h_l}) = \Psi\left(\rho_{i,k_0,h_0}, \rho_{i,k_1,h_1}, \dots, \rho_{i,k_{|D|-1},h_{|D|-1}}\right)$ is selected from $D_P^{\triangleright}$. $\Psi(\boldsymbol{h_l})$ is defined as follows:

$$\Psi\left(\rho_{i,k_0,h_0}, \rho_{i,k_1,h_1}, \dots, \rho_{i,k_{|D|-1},h_{|D|-1}}\right) = \begin{aligned} &\{\langle V_{i,k_0}^{\triangleright}, p_{i,k_0}^{\triangleright}, \hat{V}_{i,k_0}^{\triangleright}\rangle, \\ &\dots, \\ &\langle V_{i,k_{|D|-1}}^{\triangleright}, p_{i,k_{|D|-1}}^{\triangleright}, \hat{V}_{i,k_{|D|-1}}^{\triangleright}\rangle \\ &| V_{i,k_0}^{\triangleright} = \rho_{i,k_0,h_0} \wedge \\ &\dots \wedge \\ &V_{i,k_{|D|-1}}^{\triangleright} = \rho_{i,k_{|D|-1},h_{|D|-1}}\} \end{aligned} \quad (4)$$

Finally, $\Psi(\boldsymbol{h_l})$ is used to compute the value of the probabilistic data cube cell $C(D_P)[\boldsymbol{h_l}] = \langle [B_{min}^{\boldsymbol{h_l}}, B_{max}^{\boldsymbol{h_l}}], p_{\boldsymbol{h_l}}\rangle$, whose characteristic parameters $B_{min}^{\boldsymbol{h_l}}$, $B_{max}^{\boldsymbol{h_l}}$ and $p_{\boldsymbol{h_l}}$ are defined as follows (from Sect. 2, recall that in our research we focus on SUM-based data cubes):

$$B_{min}^{\boldsymbol{h_l}} = \left\lfloor \underset{\Psi(\boldsymbol{h_l}), 0 \le j \le |D|-1}{argmin} \left\{\hat{V}_{i,k_j}^{\triangleright}\right\} \right\rfloor \quad (5)$$

$$B_{max}^{\boldsymbol{h_l}} = \left\lfloor \sum_{\Psi(\boldsymbol{h_l}), j=0}^{|D|-1} \hat{V}_{i,k_j}^{\triangleright} \right\rfloor \quad (6)$$

$$p_{\boldsymbol{h_l}} = \prod_{\Psi(\boldsymbol{h_l}), j=0}^{|D|-1} p_{i,k_j}^{\triangleright} \quad (7)$$

5 Querying Probabilistic OLAP Data Cubes

Given a probabilistic $|D|$-dimensional data cube C and an input M-dimensional range-SUM query $Q = \langle L, \text{SUM}\rangle = \langle\langle [l_{k_0}, u_{k_0}], [l_{k_1}, u_{k_1}], \dots, [l_{k_{M-1}}, u_{k_{M-1}}]\rangle, \text{SUM}\rangle$ over C, such that $M \le |D|$, k_i in $\{0, 1, \dots, |D| - 1\}$, and, for each k_i, $l_{k_i} < u_{k_i}$, evaluating Q against C corresponds to compute a probabilistic answer to Q, $A^P(Q) = \langle [U_{Q,min}^P, U_{Q,max}^P], p_Q\rangle$, such that $U_{Q,min}^P < U_{Q,max}^P$ and $0 \le p_Q \le 1$ (see Sect. 2). Due to uncertainty an imprecision of multidimensional data stored in C, a *probabilistic estimator* $\Omega(Q)$ [31] must be introduced in order to provide an accurate answer to Q, $A^P(Q)$. To this end, our general approach consists in providing $A^P(Q)$ in terms of *some statistics* extracted from an *appropriate* PDF describing probabilistic multidimensional data bounded by the multidimensional range L of Q. It should be noted that a similar previous approach has been applied in the context of *querying uncertain and imprecise data streams* during past significant research efforts [13,26]. This confirms to us the authority of the approach we propose. With these ideas in mind, in order to efficiently answer range-SUM queries over probabilistic data cubes *we propose the definition of a reliable* $\langle \varepsilon, \delta \rangle$*-based probabilistic estimator over uncertain and imprecise multidimensional data*, denoted by $\langle \varepsilon, \delta \rangle$-$\Omega(Q)$, which belongs to the well-known class of $\langle \varepsilon, \delta \rangle$ *probabilistic estimators* [31], among which the *Hoeffding-based estimator* [23] is the most popular one that has been extensively used in the context of database and data-warehouse processing (e.g., [27]). Since the theory for probabilistic estimators is already available [31] and it can be directly exploited within our proposed framework for OLAP over probabilistic data, the most important research contribution we provide in this respect is represented by the *methodology for building the appropriate PDF describing uncertain and imprecise multidimensional data bounded by L* (*of Q*).

To this end, we exploit an *analytical interpretation of multidimensional data cubes* that has been proposed by us in some of our previous research experiences [14]. According to this interpretation, *MOLAP data cubes* (like the ones we consider in our research – see Sect. 2) *are viewed in terms of a collections of data rows*, which offers powerful optimization opportunities for OLAP data cube processing like the case of *OLAP data cube compression techniques* investigated in [14]. In this research, *we argue to represent range queries in terms of collections of rows, since, intuitively enough, range queries can be structurally viewed as data cubes themselves* [14]. Among other nice amenities highlighted in [14], one of the merits of the analytical interpretation of MOLAP data cubes proposed in [14] is represented by the fact it can be easily implemented within the core layer of any arbitrary OLAP server platform, due to the fact it is essentially based on the well-known *slice* (*OLAP*) *operator*, which is made available as a native operator in any OLAP server platform.

Based on this main assertion, given a range query Q, in our proposed framework for OLAP over probabilistic data we model Q as follows: $Q \equiv Q(d_i) = \langle Q_{d_i,0}, Q_{d_i,1}, \dots, Q_{d_i,|R|-1} \rangle$, such that: (*i*) $d_i \in \{0, 1, \dots, |D| - 1\}$ denotes the OLAP dimension with respect to which the slice operation is performed, called *slicing dimension*; (*ii*) $Q_{d_i,r}$, with r in $\{0, 1, \dots, |R| - 1\}$, denotes a *row-based range query* extracted from Q on the basis of such a slice operation and well-understood abstractions of *matrix theory* [19]; (*iii*) R denotes the total number of row-based range queries extracted from Q, which, in turn, depends on the selectivity of Q, denoted by $\|Q\|$. It should be noted that this proposed row-based representation of range queries is flexible enough to result independent on the number of dimensions of queries.

Let $L_{d_i,r}$ denote the multidimensional range associate to $Q_{d_i,r}$. Let $\prod_{d_i}(\mathbb{I})$ denote the *projection operator* that, given a multidimensional item $\mathbb{I}$ and a dimension d_i in the multidimensional space of $\mathbb{I}$, projects $\mathbb{I}$ with respect to d_i. Given a range-SUM query Q over a probabilistic data cube C, on the basis of the previous analytical interpretation of range queries, for each row-based range query $Q_{d_i,r}$ in $Q(d_i)$, we introduce a *discrete PDF* [31] that is obtained by *composing all the confidence intervals, and related probabilities, of* (*probabilistic*) *data cube cells in* C *contained by* $L_{d_i,r}$, denoted by $\mathcal{X}_{d_i,r}$. "Describing" the probabilistic variation of data cube cells in C contained by $L_{d_i,r}$ is the main goal of $\mathcal{X}_{d_i,r}$. Formally, $\mathcal{X}_{d_i,r}$ is defined as follows:

$$\mathcal{X}_{d_i,r}[k] = \sum_{\Pi_{d_i}(\boldsymbol{h_l}) \in L_{d_i,r}, k=0}^{|Q_{d_i,r}|-1} \langle \left[B_{min}^{\Pi_{d_i}(\boldsymbol{h_l})}, B_{max}^{\Pi_{d_i}(\boldsymbol{h_l})} \right], p_{\Pi_{d_i}(\boldsymbol{h_l})} \rangle \cdot \delta(k) \tag{8}$$

In our proposed framework for OLAP over probabilistic data, the probabilistic answer to Q, $A^P(Q)$, is retrieved according to the following 5-step methodology:

1. Fix a slicing dimension d_i among the available $|D|$ dimensions of C.
2. Based on d_i, decompose Q into R row-based range queries: $Q(d_i) = \langle Q_{d_i,0}, Q_{d_i,1}, \dots, Q_{d_i,|R|-1} \rangle$.
3. For each row-based range query $Q_{d_i,r}$ in $Q(d_i)$, compute the PDF $\mathcal{X}_{d_i,r}$ (8), thus obtaining the set (of PDF): $\mathbb{X}(Q(d_i)) = \{\mathcal{X}_{d_i,0}, \mathcal{X}_{d_i,1}, \dots, \mathcal{X}_{d_i,|R|-1}\}$.
4. From the set $\mathbb{X}(Q(d_i))$, compute the *joint PDF* [31], denoted by $\mathcal{Z}_Q^{d_i}$ – intuitively enough, $\mathcal{Z}_Q^{d_i}$ models the joint contribution of *all* the PDF associated to probabilistic multidimensional data involved by Q – this approach is inspired by [36], where PDF describing independent observations are combined for *pre-aggregation* purposes in the context of *probabilistic Data Warehouse servers*.
5. Retrieve $A^P(Q)$ as follows:

$$A^P(Q) = \int_{\langle k_0,\dots,k_{M-1}\rangle = \langle l_{k_0},\dots,l_{k_{M-1}}\rangle}^{\langle k_0,\dots,k_{M-1}\rangle = \langle u_{k_0},\dots,u_{k_{M-1}}\rangle} \mathcal{Z}_Q^{d_i}[\boldsymbol{h_l}] d\boldsymbol{h_l} \tag{9}$$

Finally, since $\mathcal{Z}_Q^{d_i}$ globally models the joint contribution of all the PDF associated to probabilistic multidimensional data involved by Q, we can *break* the dependency of $\mathcal{Z}_Q^{d_i}$ from the M-dimensionality exposed in (9), and retrieve $A^P(Q)$ by means of computing *appropriate statistics* over $\mathcal{Z}_Q^{d_i}$. From active literature [31], relevant *moments* of $\mathcal{Z}_Q^{d_i}$, i.e. *mean*, denoted by $\mathbb{E}(\mathcal{Z}_Q^{d_i})$, and *variance*, denoted by $\mathbb{V}(\mathcal{Z}_Q^{d_i})$, play the role of most appropriate statistics in this respect. In fact, according to several studies [13,26], these moments well-summarize the *statistical content* of $\mathcal{Z}_Q^{d_i}$, and are well-suited for answering OLAP queries over the involved (uncertain and imprecise) data. Based on these statistics, we finally retrieve $A^P(Q)$ as follows:

$$A^P(Q) = \|Q\| \cdot \mathbb{E}(\mathcal{Z}_Q^{d_i}) \tag{10}$$

Furthermore, the parameters ε and δ of the probabilistic estimator $\langle \varepsilon, \delta \rangle$-$\Omega(Q)$ allows us to compute a suitable confidence interval with related probability for $A^P(Q)$, i.e. expressing $A^P(Q)$ as $A^P(Q) = \langle [U_{Q,min}^P, U_{Q,max}^P], p_Q \rangle$ that is the main goal of querying probabilistic OLAP data cubes (see Sect. 2), from $\mathcal{Z}_Q^{d_i}$ directly. Since $\mathcal{Z}_Q^{d_i}$ is a joint PDF obtained from a set of PDF, it can be supposed that $\mathcal{Z}_Q^{d_i}$ follows a *quasi-Gaussian distribution* [31] (see Fig. 3). Therefore, we can exploit *numerical methods* on $\mathcal{Z}_Q^{d_i}$ to compute both parameters ε and δ of $\langle \varepsilon, \delta \rangle$-$\Omega(Q)$, and, in turn, the model

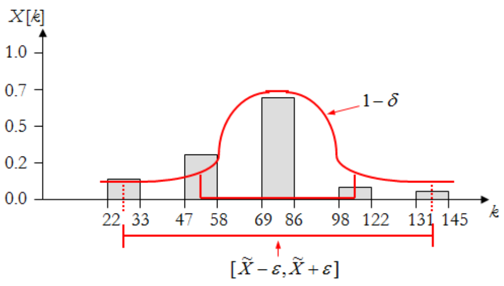


Fig. 3. An example quasi-Gaussian discrete PDF with statistical parameters ε and δ of an estimate $\tilde{X}$

parameters $U^P_{Q,min}$, $U^P_{Q,max}$ and p_Q. From active literature [31], for each probability $p = 1 - \delta$, which is equal to the area of the Gaussian bell, fixed the confidence interval $[\tilde{X} - \varepsilon, \tilde{X} + \varepsilon]$ of an estimate $\tilde{X}$, we can retrieve the corresponding value of the parameter ε, such that $P\left(\left|\tilde{X} - X\right| \leq \varepsilon\right) \geq 1 - \delta$. ε is recognized as the *accuracy* of $\langle\varepsilon, \delta\rangle$-$\Psi(Q)$ for *that* estimate $\tilde{X}$ [31]. In our proposed framework for OLAP over probabilistic data, we set $A^P(Q) \equiv \tilde{X}$ and we compute parameters ε and δ from $\mathcal{Z}^{d_i}_Q$ for the probability value $p = p_Q$ for which $\mathcal{Z}^{d_i}_Q = \|Q\| \cdot \mathbb{E}\left(\mathcal{Z}^{d_i}_Q\right)$ (i.e., (10)), denoted by ε_Q and δ_Q (note that $p_Q = 1 - \delta_Q$), respectively. Finally, we derive $U^P_{Q,min}$ and $U^P_{Q,max}$ as follows: $U^P_{Q,min} = \|Q\| \cdot \mathbb{E}\left(\mathcal{Z}^{d_i}_Q\right) - \varepsilon_Q$ and $U^P_{Q,max} = \|Q\| \cdot \mathbb{E}\left(\mathcal{Z}^{d_i}_Q\right) + \varepsilon_Q$, respectively, which authoritatively complete our query model for probabilistic OLAP data cubes.

6 Conclusions and Future Work

A complete framework for computing and querying multidimensional OLAP data cubes over probabilistic data has been proposed in this paper. We also conducted a set of preliminary experiments over synthetic probabilistic datasets (not shown in this paper for space reasons) that have confirmed the feasibility of the proposed framework. Future work is mainly focused on developing and conducting a wide experimental campaign on both synthetic and real-life probabilistic datasets, and on extending the framework in order to make it able of dealing with more complex SQL aggregation operators beyond the simple ones considered in the actual research (e.g., SUM, COUNT) and domain constraints over dataset attributes like those considered in [8].

References

[1] Agarwal, S., Agrawal, R., Deshpande, P., Gupta, A., Naughton, J.F., Ramakrishnan, R., Sarawagi, S.: On the Computation of Multidimensional Aggregates. In: Proceedings of VLDB 1996 Int. Conf. (1996)

[2] Agrawal, P., Benjelloun, O., Sarma, A.D., Hayworth, C., Nabar, S.U., Sugihara, T., Widom, J.: Trio: A System for Data, Uncertainty, and Lineage. In: Proceedings of VLDB 2006 Int. Conf. (2006)

[3] Barbarà, D., Garcia-Molina, H., Porter, D.: The Management of Probabilistic Data. IEEE Transactions on Knowledge Data Engineering 4(5) (1992)

[4] Benjelloun, O., Sarma, A.D., Halevy, A.Y., Theobald, M., Widom, J.: Databases with Uncertainty and Lineage. VLDB Journal 17(2) (2008)

[5] Bonnet, P., Gehrke, J.E., Seshadri, P.: Towards Sensor Database Systems. In: Proceedings of ACM MDM Int. Conf. (2001)

[6] Burdick, D., Deshpande, P.M., Jayram, T.S., Ramakrishnan, R., Vaithyanathan, S.: OLAP over Uncertain and Imprecise Data. In: Proceedings of VLDB 2005 Int. Conf. (2005)

[7] Burdick, D., Deshpande, P.M., Jayram, T.S., Ramakrishnan, R., Vaithyanathan, S.: Efficient Allocation Algorithms for OLAP over Imprecise Data. In: Proceedings of VLDB 2006 Int. Conf. (2006)

[8] Burdick, D., Doan, A., Ramakrishnan, R., Vaithyanathan, S.: OLAP over Imprecise Data with Domain Constraints. In: Proceedings of VLDB 2007 Int. Conf. (2007)

[9] Chen, A.L.P., Chiu, J.-S., Tseng, F.S.-C.: Evaluating Aggregate Operations over Imprecise Data. IEEE Transactions on Knowledge Data Engineering 8(2) (1996)

[10] Cheng, R., Kalashnikov, D., Prabhakar, S.: Evaluating Probabilistic Queries over Imprecise Data. In: Proceedings of ACM SIGMOD 2003 Int. Conf. (2003)

[11] Cheng, R., Singh, S., Prabhakar, S., Shah, R., Vitter, J.S., Xia, Y.: Efficient Join Processing over Uncertain Data. In: Proceedings of ACM CIKM 2006 Int. Conf. (2006)

[12] Colliat, G.: OLAP, Relational, and Multidimensional Database Systems. SIGMOD Record 25(3) (1996)

[13] Cormode, G., Garofalakis, M.: Sketching Probabilistic Data Streams. In: Proceedings of ACM SIGMOD 2007 Int. Conf. (2007)

[14] Cuzzocrea, A.: Improving Range-Sum Query Evaluation on Data Cubes via Polynomial Approximation. Data & Knowledge Engineering 56(2) (2006)

[15] Cuzzocrea, A., Wang, W.: Approximate Range-Sum Query Answering on Data Cubes with Probabilistic Guarantees. Journal of Intelligent Information Systems 28(2) (2007)

[16] Dalvi, N., Suciu, D.: Efficient Query Evaluation on Probabilistic Databases. In: Proceedings of VLDB 2004 Int. Conf. (2004)

[17] Dalvi, N., Suciu, D.: Management of Probabilistic Data: Foundations and Challenges. In: Proceedings of ACM PODS 2007 Int. Conf. (2007)

[18] Deligiannakis, A., Roussopoulos, N.: Extended Wavelets for Multiple Measures. In: Proceedings of ACM SIGMOD 2003 Int. Conf. (2003)

[19] Golub, G.H., Van Loan, C.F.: Matrix Computation, 2nd edn. Johns Hopkins University Press, Baltimore (1989)

[20] Gray, J., Chaudhuri, S., Bosworth, A., Layman, A., Reichart, D., Venkatrao, M., Pellow, F., Pirahesh, H.: Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals. Data Mining and Knowledge Discovery 1(1) (1997)

[21] Han, J., Kamber, M.: Data Mining: Concepts and Techniques, second ed. Morgan Kauffmann Publishers, San Francisco (2006)

[22] Harinarayan, V., Rajaraman, A., Ullman, J.: Implementing Data Cubes Efficiently. In: Proceedings of ACM SIGMOD 1996 Int. Conf. (1996)

[23] Hellerstein, J.M., Haas, P.J., Wang, H.J.: Online Aggregation. In: Proceedings of ACM SIGMOD 1997 Int. Conf. (1997)

[24] Ho, C.-T., Agrawal, R., Megiddo, N., Srikant, R.: Range Queries in OLAP Data Cubes. In: Proceedings of ACM SIGMOD 1997 Int. Conf. (1997)

[25] Hua, M., Pei, J., Zhang, W., Lin, X.: Ranking Queries on Uncertain Data: A Probabilistic Threshold Approach. In: Proceedings of ACM SIGMOD 2008 Int. Conf. (2008)

[26] Jayram, T.S., McGregor, A., Muthukrishnan, S., Vee, E.: Estimating Statistical Aggregates on Probabilistic Data Streams. In: Proceedings of ACM PODS 2007 Int. Conf. (2007)

[27] Jin, R., Glimcher, L., Jermaine, C., Agrawal, G.: New Sampling-Based Estimators for OLAP Queries. In: Proceedings of IEEE ICDE 2006 Int. Conf. (2006)

[28] Kimelfeld, B., Sagiv, Y.: Maximally Joining Probabilistic Data. In: Proceedings of ACM PODS 2007 Int. Conf. (2007)

[29] Lian, X., Chen, L.: Probabilistic Ranked Queries in Uncertain Databases. In: Proceedings of EDBT 2008 Int. Conf. (2008)

[30] McClean, S.I., Scotney, B.W., Shapcott, M.: Aggregation of Imprecise and Uncertain Information in Databases. IEEE Transactions on Knowledge Data Engineering 13(6) (2001)

[31] Papoulis, A.: Probability, Random Variables, and Stochastic Processes, second ed. McGraw-Hill, New York (1984)

[32] Ré, C., Suciu, D.: Approximate Lineage for Probabilistic Databases. PVLDB 1(1) (2008)

[33] Ross, R., Subrahmanian, V.S., Grant, J.: Aggregate Operators in Probabilistic Databases. Journal of the ACM 52(1) (2005)

[34] Sarma, A.D., Theobald, M., Widom, J.: Exploiting Lineage for Confidence Computation in Uncertain and Probabilistic Databases. In: Proceedings of IEEE ICDE Int. Conf. (2008)

[35] Soliman, M.A., Ilyas, I.F., Chang, K.C.-C.: Top-K Query Processing in Uncertain Databases. In: Proceedings of IEEE ICDE 2007 Int. Conf. (2007)

[36] Timko, I., Dyreson, C.E., Pedersen, T.B.: Pre-Aggregation with Probability Distributions. In: Proceedings of ACM DOLAP 2006 Int. Conf. (2006)

[37] Vassiliadis, P., Sellis, T.: A Survey of Logical Models for OLAP Databases. SIGMOD Record 28(4) (1999)

[38] Yi, K., Li, F., Srivastava, D.: Kollios. G.: Efficient Processing of Top-K Queries in Uncertain Databases. In: Proceedings of IEEE ICDE 2008 Int. Conf. (2008)

Correcting Missing Data Anomalies with Clausal Defeasible Logic

Peter Darcy, Bela Stantic, and Abdul Sattar

Institute for Integrated and Intelligent Information Systems
Griffith University
{P.Darcy,B.Stantic,A.Sattar}@griffith.edu.au

Abstract. Databases are used globally to store essential information required for various business applications such as automated data capturing. Unfortunately, due to missing record anomalies present within the repository, the overall integrity of stored information is compromised. Currently, filtration and rule-based techniques have been proposed to correct the problem, but due to a lack of high-level reasoning, ambiguous scenarios lead to anomalies persisting within the database. In this paper, we propose an enhanced *Non-Monotonic Reasoning* cleaning architecture that utilises intelligent analysis coupled with *Clausal Defeasible Logic* to rectify the missing data by generating and restoring imputed data. From our experimental evaluation, we have found that our proposed technique surpasses other leading intelligence classifiers such as Bayesian and Neural Networks.

1 Introduction

False negative anomalies are missing gaps of information that are meant to be present within databases. This problem is significant due to its potential to lower the accuracy and quality of the entire data set resulting in further data processes being hindered. Missing data anomalies are most persistent within automatic data capturing technology that utilises hardware to record information. Without high level intelligence designed to correct false negative anomalies, data sets that store information acquired from technologies such as automated recording devices will continue to process incorrect information hindering the applications it was designed for.

Radio Frequency Identification (RFID) is data capturing technology which automatically records data from tags that respond to readers. Of the problems found within passive RFID systems, ambiguous false negative observations remain the hardest to correct [1]. To correct the problem of missing observations, past approaches have utilised filtering algorithms and rule-based correction techniques to prevent the false negatives where possible. Unfortunately, within ambiguous situations in which the correction method is not easily determined, these methodologies fail to clean the data set adequately. The problems hindering these approaches include the fact that there is a lack of analytical information and intelligence when cleaning the data set with filters and rules respectively.

In this paper, we propose a methodology that cleans the stored data set using a high level intelligence reasoning engine. The system utilises analytical information generated from the missing observation coupled with Non-Monotonic Reasoning engines to correctly establish the likeliest set of data to insert back into the database. We specifically employ Clausal Defeasible Logic (CDL) as our Non-Monotonic Reasoning approach to arrive at a conclusion. We believe that by cleaning the data at a deferred stage allows the cleaning algorithm to have access to enough information needed to correctly impute the correct results. Additionally, by incorporating a high level intelligence algorithm such as Non-Monotonic Reasoning, we ensure the maximum likelihood to decipher highly ambiguous situations.

To evaluate the performance of our proposed system, we have conducted two experiments. The first was designed to assess which CDL formulae provided the highest cleaning rate, while the second evaluated the performance of our approach against Bayesian and Neural Network approaches. From our experiments, we have found the highest performing Clausal Defeasible Logic formula. With regards to our significance investigation, we have demonstrated that our proposed concept achieves the highest average cleaning rate when compared to Bayesian and Neural Networks under the same conditions.

The remainder of this paper is structured as follows: Section 2 will deliver background information relating to RFID and Non-Monotonic Reasoning crucial to understanding our proposed concept. A concise description of previous and related work will be provided in Section 3 shortly before we explore our own methodology in Section 4. An evaluation of our experimentation will be provided in Section 5 followed by our results and analysis in Section 6. Finally, Section 7 will discuss conclusions we have drawn from the our proposed methodology and future work we intend to investigate following the research we have conducted.

2 Background

RFID systems refer to a system that automatically identifies multiple amounts of tagged objects within a certain proximity to a reader. Although the potential benefits of cheap and durable passive tagging architectures are great, ambiguous anomalies such as missed readings hinder the world-wide adoption of this technology. High level intelligence engines, such as Non-Monotonic reasoning, may be harnessed to provide the resolution to highly ambiguous scenarios such as missing data to increase the integrity of the data set. Non-Monotonic reasoning approaches such as defeasible logic and clausal defeasible logic are employed to determine the optimal conclusion when given ambiguous information as input.

2.1 Radio Frequency Identification

Radio Frequency Identification (RFID) utilises radio transmissions between a reader and identifying tags to wirelessly locate objects automatically. Items are fitted with tags which are interrogated by readers resulting in the return of the unique Electronic Product Code (EPC) [2]. Unfortunately, there are several issues associated with the RFID architecture, specifically the passive tag system. There are two persistent anomalies found within recorded data sets that are continually introduced. This may be attributed to various factors such as collision, detuning or water/metallic interference among tags [3]. These anomalies are false positive readings where the data captured did not exist in reality, and false negative readings where data is not present in the data set but is required to be. Of these two anomalies, false negative errors may be considered the hardest to correct as the data is not recorded into the database minimising the contextual information needed to correctly impute what readings may have originally been present. It has been estimated that only 60%-70% of recordings have been estimated to be captured within any RFID architecture [4].

2.2 Non-monotonic Reasoning

Non-Monotonic Reasoning utilises logic to eliminate potential answers to a given problem based on available information until a single conclusion has been determined. When given the option to arrive at a conclusion, it will utilise present information to either prove or disprove an ambiguous situation with high level intelligence. An example in which Non-Monotonic Reasoning would decipher the correct solution would be when determining if Siberian Huskies bark. It is generally accepted that all dogs inherently bark, however, the Siberian Huskies are one of a few rare breeds of dogs which do not [5].

Clausal Defeasible Logic (CDL) is software designed to incorporate the logic found in Non-Monotonic Reasoning within a computational environment and to be integrated into various applications [6]. The benefit of embedding Clausal Defeasible Logic is that the engine will deterministically arrive at various intelligent conclusions due to the use of different levels of ambiguous strengths. The scenario in which Siberian Huskies do not bark has been represented within the logical map utilised in CDL as pictured in Figure 1. As seen in the logic map, the specific rule that Siberian Huskies do not bark outweighs the general rule in which dogs bark is represented with an arch that favours rules which are further anti-clockwise. In most cases, the logic arrives at the general conclusion, however this may be beaten by the specific rules that defy the previously established

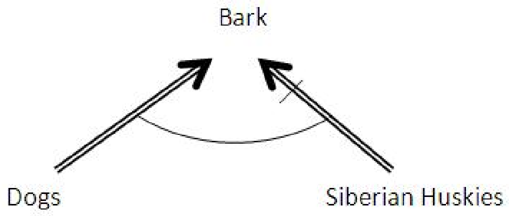


Fig. 1. The Logic Map that houses the clausal defeasible rules used when deciding if the Siberian Huskies will bark due to it being a dog or not

deduction. To properly find the correct conclusion, there are five core ambiguity strengths [7] which include the following:

- μ**:** The strongest of the formulae which will only prove the conclusion with factual information.
- α**:** A formula in which any conjunction of the π and β formulae are used to reach its conclusion.
- π**:** The formula in which ambiguity is propagated to reach its conclusion.
- β**:** The formula in where ambiguity will not be allowed to be used to draw its conclusions.
- δ**:** The disjunction of π and β are used to draw conclusions.

3 Related Work

Passive RFID anomalies has been studied extensively within related work with two main methods used to correct the missed readings. The first methodology uses filtering algorithms to correct incoming records when data is first captured. The filtration processes rely on anti-collision protocols to correct tagged observations as they are being physically read pre-data storage [8]. The second approach applies correcting algorithms to the stored data, which enhances the integrity of the observations within the data storage. This approach employs user-specified rules to modify the recorded observations in an attempt to enhance the integrity post-data storage [9].

While the proposed approaches can correct simple false-negative anomalies, highly ambiguous missing data may not be recovered due to the complex nature of the observations. We believe that due to flaws such as low level of analytical information and lack of high level intelligence, the filtration and rule-based methodologies will not be able to handle scenarios in which missing data is not easily replaced. With regards to the filtration approach, the cleaning process is performed at the edge which will only review the present and past information passed to the middleware. This results in the correction not taking future observations into account and has been compared with a Bayesian Network in previous literature [10]. In contrast, the rule-based approach utilises user-specified rules that are applied to the data set after the readings have been stored. However, logical anomalies may be introduced unknowingly in the event where false negative anomalies have a huge level of ambiguity. Previous research has found that by adding a higher level intelligence such as Plausible Logic improves upon the framework [11].

4 Methodology

In this paper, we propose the use of an advanced data analysis methodology coupled with high level intelligence to correctly decipher the likeliest candidates of observations to be returned into the data set. In the following section, we will outline the motivation and scenario considered in this work followed by a description of the system architecture of our approach. These discussions will be followed by the database structure that houses the RFID observations and all assumptions made towards our methodology.

4.1 Motivation and Scenario

False Negative anomalies are hazardous to all applications that utilise RFID as it prevents the recording of data lowering the overall integrity of the whole data set. Within previous work, we have established that there is large potential for the fusion of thorough analysis coupled with high level intelligence to adequately replace missed readings. Additionally, we have also put forth a preliminary analysis of a simplistic CDL logic engine using high level analysis which resulted in high cleaning rates. We have since decided to enhance the cleaning rules using different Non-Monotonic Reasoning software and comparing these against an already state-of-the-art approach.

The scenario upon which we have tailored our approach would include an enclosed static environment where tracking items is essential. Missed readings, however, may occur in this approach which need to be corrected before the data can be utilised in meaningful contexts. The most beneficial scenario would include false negative readings to occur randomly and rarely in which case it would be easier to correct. Our concept has been focused to provide higher intelligence for case studies in which missed readings occur consecutively.

For example, within a stocking warehouse that transfers stocked items via conveyer belt to various locations of the environment, such as the mock environment depicted in Figure 2. The circles within the diagram represent the readers and their respective ranges. The box with T1 inside represents the stocked item being tracked as Tag 1 and the dotted line represents the path of which T1 would travel in this scenario. If the items come too close within proximity or the tag is facing the opposite direction to the reader, the stock's tags may not be observed at certain locations.

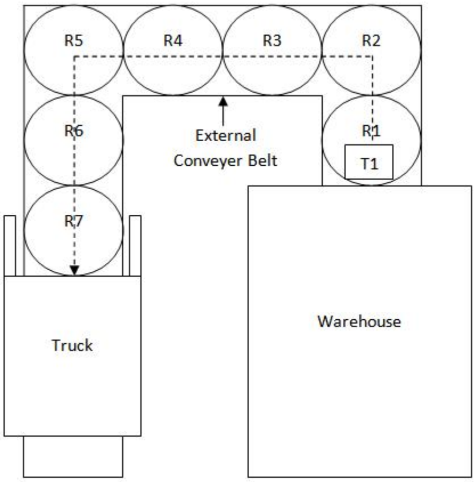


Fig. 2. A graphical representation of our ideal scenario in which a RFID-enabled conveyer belt tracks items from a warehouse to a truck

Table 1. A Table housing our ideal sample scenario RFID captured data.

EPC	Reader ID	Timestamp
T1	R1	2009-11-24 10:30:00.000
T1	R2	2009-11-24 10:31:00.000
T1	R3	2009-11-24 10:32:00.000
T1	R7	2009-11-24 10:26:00.000

Within this scenario, there may be a situation in which the insertion of the observation may not be straight forward. Within our sample scenario previously mentioned, and the sample data generated within Table 1, we can see that *T1* is read at readers *R1*, *R2*, *R3* *R7*. From this information, conventional data correction algorithms would either replace all readings between *R3* and *R7* with the reader location *R3* or discard *R7* as a false positive anomaly. However, our approach would instead derive the shortest path from the available map data and insert it within the data set, thereby increasing the integrity of the overall information of *T1*.

4.2 System Architecture

As seen in Figure 3, we have divided our system's architecture into three core components. The first is designed to analyse the data where the missed reading occurred which we have named the *Analysis Phase*. The data discovered in this Analysis Phase is then passed onto the *Intelligence Phase* where the correct permutation is selected. After the resulting data set has been chosen, the *Loading Phase* will complete the program's cycle by inputting the information back into the data warehouse.

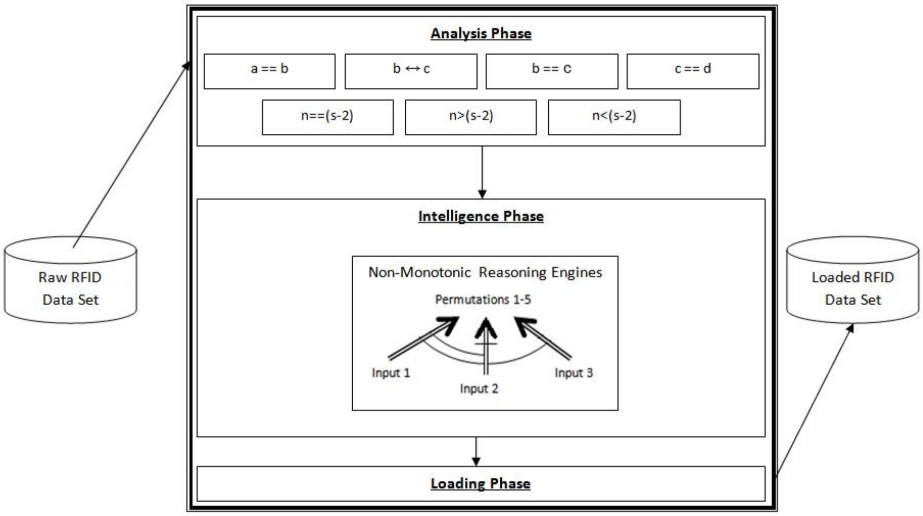


Fig. 3. A high level visualisation of inner processes used within our approach

Analysis Phase. The analysis phase consists of our tool locating missed readings and then discovering essential data about the anomaly. The first process is to divide the tags into "Tag Streams" as seen in Figure 4. These tag streams include chronical information relating only to one individual tag. From these tag streams, certain information is ascertained relating to the nature of the false negative anomaly. This includes finding the reader locations of the observations two readings before and directly before the anomaly (a and b respectively), directly after and two readings after the reader (c and d respectively). Additionally, the shortest path between readings b and c using the map data is found. The total missing readings calculated via the number of missing timestamps (n), and the amount of observations within the shortest path (s), is then calculated. From this information, we ascertain the following analytical data:

- $a == b$
- $b \leftrightarrow c$
- $b == c$
- $d == c$
- $n == (s - 2)$
- $n > (s - 2)$
- $n > (s - 2)$

We utilise four main arithmetic operations to obtain these binary analytical information. These include the equivalent symbol $==$, the less $<$ and greater than $>$ symbols, and the $\leftrightarrow$ symbol we have elected to represent geographical proximity. The reason as to why s is always having two taken away from it is that the shortest path always includes the boundary readers b and c which are not included within the n calculation.

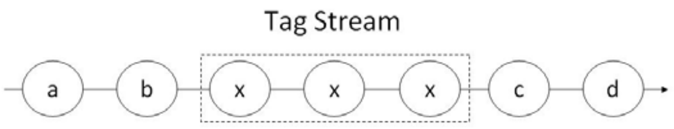


Fig. 4. A visual representation of how we analyse one tag at a given moment within a Tag Stream

Intelligence Phase. The intelligence phase is when the various permutations of the missing data are generated as candidates to be restored in the data set. The five different permutations that are been generated depicted in Figure 5 are described below:

- Permutation 1: All missing values are replaced with the reader location of observation b.
- Permutation 2: All missing values are replaced with the reader location of observation c.

– Permutation 3: The shortest path is slotted into the middle of the missing data gap. Any additional missing gaps on either end of the shortest path are substituted with values b for the left side, and c for the right.
– Permutation 4: The shortest path is slotted into the latter half of the missing data gap. Any additional missing gaps on the former end are substituted with value b.
– Permutation 5: As the anti-thesis of Permutation 4, the shortest path is slotted into the former half of the missing data gap. Additional missing gaps found at the latter end of the missing data gap are substituted with value c.

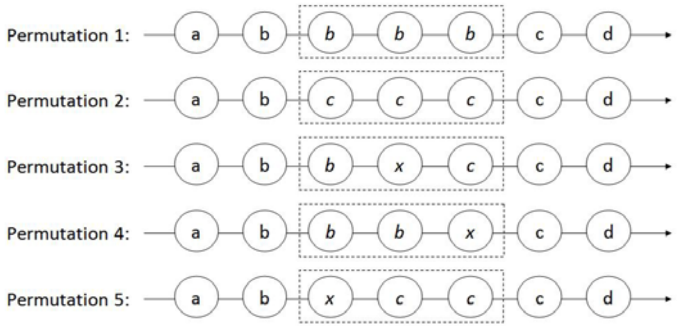

Permutation 1: a b b b b c d
Permutation 2: a b c c c c d
Permutation 3: a b b x c c d
Permutation 4: a b b b x c d
Permutation 5: a b x c c c d

Fig. 5. An illustration of what reader values are placed into each false negative anomaly for each of the five different permutations. Please note that the shortest path is represented as x.

After the data analysis and permutation formations are completed, all relevant information is passed into the *Non-Monotonic Reasoning Engine*. The engine will then return an answer deterministically as to which permutation best suits the missing gap of data, based on the rules we have created shown in Tables 2 - 6. Each of the rules present within the tables are combinations found from the analytical data joined by "and" statements $\bigwedge$) which have been gathered within the Analysis Phase. Within the logic engine build, the precedence of the rules correspond to the larger number of the rule (for example, rule 17 will beat rule 4 in Table 3. In the event that more than one permutation has been found to be ideal in a given situation, we use the following hierarchical weighting: Permutation 3 > Permutation 1 > Permutation 2 > Permutation 4 > Permutation 5. In the unlikely case where no conclusions have been drawn from the *Non-Monotonic Reasoning Engine*, Permutation 3 will be elected as the default candidate due to it having perfect symmetry within the imputed data. This ordering has been configured to be the most accurate conclusion assuming that the amount of consecutive missed readings are low due to the randomness of the anomalies.

Table 2. A Table depicting the Non-Monotonic Reasoning rules used to create the Permutation 1 logic engine

Rule No.	Rule	Conclusion
1	b==c	∼perm1
2	b==c⋀n<(s-2)	∼perm1
3	b==c⋀n==(s-2)	∼perm1
4	a==b	perm1
5	c==d⋀n==(s-2)	∼perm1
6	c==d⋀n>(s-2)	∼perm1
7	c==d⋀b↔c⋀n==(s-2)	∼perm1
8	c==d⋀b↔c⋀n>(s-2)	∼perm1
9	a==b⋀b↔c	perm1
10	a==b⋀b↔c⋀n==(s-2)	perm1
11	a==b⋀b↔c⋀b==c⋀∼c==d⋀n==(s-2)	perm1
12	c==d	∼perm1
13	c==d⋀b↔c	∼perm1
14	a==b⋀b↔c⋀n>(s-2)	perm1
15	a==b⋀b==c⋀b↔c⋀∼c==d⋀n>(s-2)	perm1
16	∼b↔c	∼perm1
17	n<(s-2)	∼perm1

Table 3. A Table depicting the Non-Monotonic Reasoning rules used to create the Permutation 2 logic engine

Rule No.	Rule	Conclusion
1	b==c	∼perm2
2	b==c⋀n<(s-2)	∼perm2
3	b==c⋀n==(s-2)	∼perm2
4	c==d	perm2
5	a==b⋀n==(s-2)	∼perm2
6	a==b⋀n>(s-2)	∼perm2
7	a==b⋀b↔c⋀n==(s-2)	∼perm2
8	a==b⋀b↔c⋀n>(s-2)	∼perm2
9	b↔c⋀c==d	perm2
10	b↔c⋀c==d⋀n==(s-2)	perm2
11	∼a==b⋀b↔c⋀b==c⋀c==d⋀n==(s-2)	perm2
12	a==b	∼perm2
13	a==b⋀b↔c	∼perm2
14	b↔c⋀c==d⋀n>(s-2)	perm2
15	∼a==b⋀b==c⋀b↔c⋀c==d⋀n>(s-2)	perm2
16	∼b↔c	∼perm2
17	n<(s-2)	∼perm2

Table 4. A Table depicting the Non-Monotonic Reasoning rules used to create the Permutation 3 logic engine

Rule No.	Rule	Conclusion
1	$a==b \bigwedge c==d$	perm3
2	$\sim a==b \bigwedge \sim c==d$	∼perm3
3	$a==b \bigwedge c==d \bigwedge n>(s-2)$	perm3
4	$\sim a==b \bigwedge \sim c==d \bigwedge \sim n>(s-2)$	∼perm3
5	$n==(s-2)$	perm3
6	$b==c$	∼perm3
7	$a==b \bigwedge c==d \bigwedge n==(s-2)$	perm3
8	$n<(s-2)$	∼perm2

Table 5. A Table depicting the Non-Monotonic Reasoning rules used to create the Permutation 4 logic engine

Rule No.	Rule	Conclusion
1	$c==d$	∼perm4
2	$b==c$	∼perm4
3	$a==b$	perm4
4	$\sim a==b$	∼perm4
5	$n>(s-2) \bigwedge a==b$	perm4
6	$\sim n>(s-2)$	∼perm4
7	$\sim a==b \bigwedge \sim n>(s-2)$	∼perm4

Table 6. A Table depicting the Non-Monotonic Reasoning rules used to create the Permutation 5 logic engine

Rule No.	Rule	Conclusion
1	$a==b$	∼perm5
2	$b==c$	∼perm5
3	$c==d$	perm5
4	$\sim c==d$	∼perm5
5	$n>(s-2) \bigwedge c==d$	perm5
6	$\sim n>(s-2)$	∼perm5
7	$\sim c==d \bigwedge \sim n>(s-2)$	∼perm5

Loading Phase. The loading phase consists of the selected permutation being uploaded back into the data storage at the completion of the *Intelligence Phase*. The user will have the opportunity to either elect to load the missing data into the current data repository, or copy the entire data set and only modify the copied data warehouse. This option would effectively allow the user to revisit the original data set in the event that the restored data is not completely accurate.

4.3 Database Structure

To store the information recorded from the RFID reader, we utilise portions of the "Data Model for RFID Applications" DMRA database structure found in Siemens Middleware software [12]. Additionally, we have introduced a new table called MapData designed to store the map data crucially needed within our application. Within the MapData table, two Reader IDs are stored in each row to dictate if the two readers are geographically within proximity.

4.4 Assumptions

We have made three assumptions that are required for the entire process to be completed. The first assumption is that the data recorded will be gathered periodically. The second assumption we presume within our scenario is that the amount of time elected for the periodic readings is less than the amount of physical time needed to move from one reader to another. This is important as we base our methodology around the central thought that the different readings will not skip over readers that are geographically connected according to the mapdata. The final assumption we make is that all readers and items required to be tracked will be enclosed in a static environment that has readers which cover the tracking area.

5 Experimental Evaluation

Within the following section, we have included a thorough description of the setup of the experimentation used in our methodology. First, we discuss the environment used to house the programs. This is followed by a detailed discussion of our experimentation including the two experiments we performed and their respective data sets used.

5.1 Environment

Our methodology has been coded in the C++ language and compiled with Microsoft Visual Studio C++. The code written to derive the lookup table needed for the Non-Monotonic Reasoning data has been written in Haskell and compiled using Cygwin Bash Shell. All programs were written and executed on Dell machine with Windows XP Service Pack 3 operating system installed.

5.2 Experiments

We have conducted two experiments to adequately measure the performance of our methodology. The first experiment conducted was to determine which of the clausal defeasible logic formulae performs most successfully when attempting to correct large amounts of scenarios. The training cases used in the first experiment consisted of three sets of data consisting of 100, 500 and 1,000 ambiguous false negative anomaly cases. These three sets of data is used to represent the 60% - 70% anomalies of a small, medium and large database respectively.

The second experiment was designed to test the performance of our selected highest performing Non-Monotonic Reasoning logic against both a bayesian and neural Network approach. The reason as to why these techniques were selected as opposed to other related work is that only other state-of-the-art classifying techniques can be compared in respect to seeking the select solution from a highly ambiguous situation.

The second experiment testing sets included four data repositories consisting of 500, 1,000, 5,000 and 10,000 ambiguous false negative anomaly cases. We defined our scoring system as if the respective methodologies were able to return the correct permutation of data that had been previously defined. All data within the training and testing set have been simulated to emulate real RFID observational data.

6 Results and Analysis

To thoroughly test our application, we devised two different examinations which we have labelled the *Non-Monotonic Reasoning* and *Significance Experiments.* The Non-Monotonic reasoning experiment compared the cleaning rate of each of the *Clausal Defeasible Logic* formulae. The highest performing Non-Monotonic reasoning setup was then compared to Bayesian and Neural network approaches to demonstrate the significance of our cleaning algorithm.

6.1 Non-monotonic Reasoning Experiment

We created the first experiment with the goal of determining which of the five CDL formulae would be able to clean the highest rate of highly ambiguous missing RFID observations. We did this by comparing the cleaning results of the μ, α, π, β and δ formulae on various training cases. There were three training sets in all with 100, 500 and 1,000 ambiguous false negative anomalies. Additionally, at the completion of these experiments, the average was determined for all three test cases and was used to ascertain which of the five formulae would be used within the Significance Experiment.

As seen in Figure 6, the highest average achieving formula has been found to be α (Alpha). This is probably due to the fact that it discovers cases in which both the β and π formulae agree upon, increasing the intelligence of the decision. Also of note is that the disjunction of β and π formulae shown within δ achieves a relatively high average cleaning rate also. The lowest performing average cleaning rate has been found to be β, which is probably due to its non-acceptance of ambiguity when drawing its conclusion. We believe it is crucial for the cleaner to have a low level of ambiguity when drawing its conclusions as the problem of missed readings needs a level probability to infer what readings need to be replaced. As stated above, we have chosen the α formula as the highest performing cleaner to be used within the Significance Experiment.

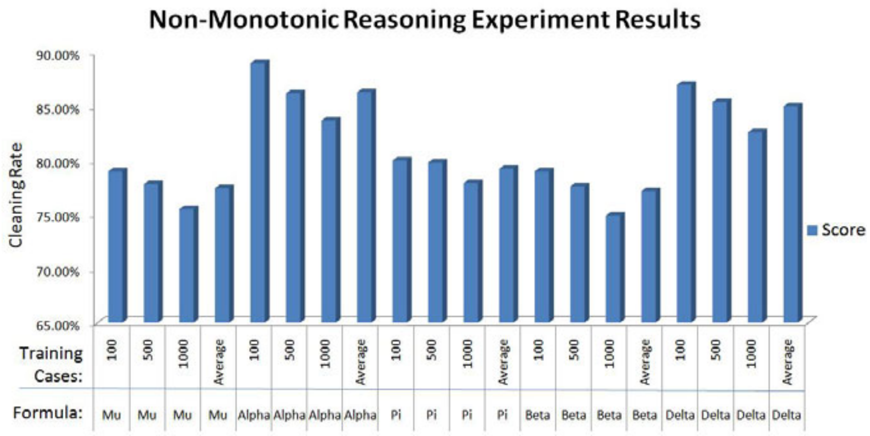


Fig. 6. The results of the Non-Monotonic Reasoning Experiment where the cleaning rate has been found for the five CDL formulae μ (Mu), α (Alpha), π (Pi), β (Beta) and δ (Delta)

6.2 Significance Experiment

The goal of our second experimental evaluation was designed to put three classifiers through a series of test cases with large amounts of ambiguous missing observations. The three different classifications techniques included our Non-Monotonic

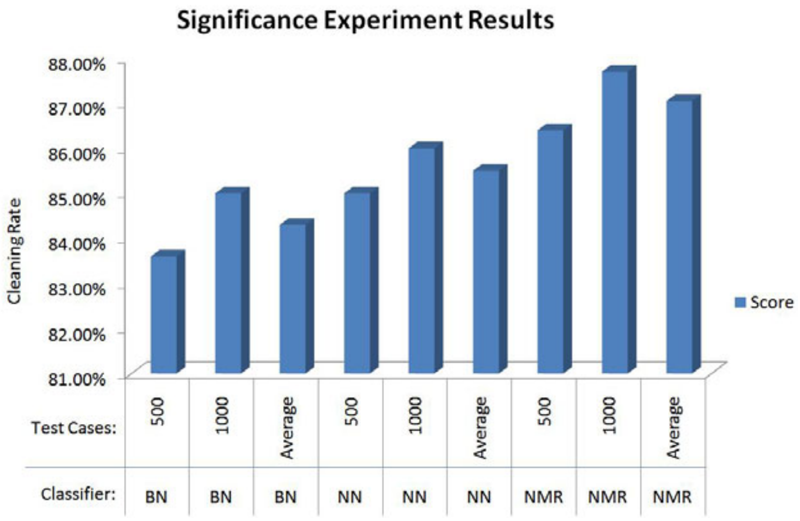


Fig. 7. The results of the Significance Experiment where the cleaning rate has been found for the Bayesian Network, Neural Network and Non-Monotonic Reasoning using the α formula

Reasoning engine with CDL using the α compared against both Bayesian and Neural Networks.

Both of the networks were trained using a Genetic Algorithm using 10 chromosomes with a single-point crossover function bred for 50 generations. We designed the experiment to have an abnormally high amount of ambiguous false negative anomalies consisting of 500 and 1,000 test cases to thoroughly evaluate each approach. After these experiments had concluded, we derived the average of each technique to find the highest performing classifier.

The results of this experiment, depicted in Figure 7, has shown that on average, our Non-Monotonic Reasoning approach achieved the highest cleaning rate. It is important to note that it achieved the highest results when cleaning the data set with 1,000 test cases. We believe that the high results may be due to the deterministic nature of our methodology preventing erroneous permutations being selected. In contrast, the probabilistic methodologies have a higher chance of selecting incorrect imputed values which in turn produces artificial false positive anomalies and not recovering the original missing data.

7 Conclusion

In this paper, we have proposed the utilisation of intelligent Clausal Defeasible Logic engines to clean highly ambiguous false negative RFID anomaly cases. Through experimental studies we have identified that our concepts performs most effectively when used to restore consecutive missed observations. We have also conducted experiments to demonstrate the significance of our approach compared to other state-of-the-art technologies. More specifically this work makes the following contributions to the field:

– We put forth an enhanced logic engine and showed that it can deal with ambiguous false negative RFID observation anomalies.
– We found that the highest performing Clausal Defeasible Logic formula is α.
– We compared our proposed concept to other leading state-of-the-art classification techniques and found that our approach obtains a higher cleaning rate.

We have specifically tailored our approach to be compatible with RFID technology. However, this concept may be applied to other applications that have missing spatio-temporal observational data. With regards to future work, it would be interesting to investigate "Would the increase of complexity in the analysis phase result in a higher cleaning rate?" Specifically, increasing the amount of observational data collected in the analysis phase. We would also like to apply our methodology to a practical RFID supply chain and other applications that would benefit from our concept.

Acknowledgment

This research is partly sponsored by ARC (Australian Research Council) grant no. DP0557303.

References

1. Derakhshan, R., Orlowska, M.E., Li, X.: RFID Data Management: Challenges and Opportunities. In: RFID 2007, pp. 175–182 (2007)
2. Chawathe, S.S., Krishnamurthy, V., Ramachandran, S., Sarma, S.E.: Managing RFID Data. In: VLDB, pp. 1189–1195 (2004)
3. Floerkemeier, C., Lampe, M.: Issues with RFID usage in ubiquitous computing applications. In: Ferscha, A., Mattern, F. (eds.) PERVASIVE 2004. LNCS, vol. 3001, pp. 188–193. Springer, Heidelberg (2004)
4. Jeffery, S.R., Garofalakis, M.N., Franklin, M.J.: Adaptive Cleaning for RFID Data Streams. In: VLDB, pp. 163–174 (2006)
5. Lam, B.: NICTA: SPINdle. NICTA (2009) `http://spin.nicta.org.au/spindleOnline/demo.html` (accessed: October 25, 2009)
6. Billington, D.: Propositional Clausal Defeasible Logic. In: Hölldobler, S., Lutz, C., Wansing, H. (eds.) JELIA 2008. LNCS (LNAI), vol. 5293, pp. 34–47. Springer, Heidelberg (2008)
7. Billington, D.: An Introduction to Clausal Defeasible Logic. David Billington's Home Page (August 2007), `http://www.cit.gu.edu.au/~db/research.pdf` (accessed: July 3, 2008)
8. Shih, D.H., Sun, P.L., Yen, D.C., Huang, S.M.: Taxonomy and Survey of RFID Anti-Collision Protocols. Computer Communications 29(11), 2150–2166 (2006)
9. Rao, J., Doraiswamy, S., Thakkar, H., Colby, L.S.: A Deferred Cleansing Method for RFID Data Analytics. In: VLDB, pp. 175–186 (2006)
10. Darcy, P., Stantic, B., Sattar, A.: Improving the Quality of RFID Data by Utilising a Bayesian Network Cleaning Method. In: Proceedings of the IASTED International Conference Artificial Intelligence and Applications (AIA 2009), pp. 94–99 (2009)
11. Darcy, P., Stantic, B., Derakhshan, R.: Correcting Stored RFID Data with Non-Monotonic Reasoning. Principles and Applications in Information Systems and Technology (PAIST) 1(1), 65–77 (2007)
12. Liu, S., Wang, F., Liu, P.: A Temporal RFID Data Model for Querying Physical Objects. Technical Report TR-88, TimeCenter (2007)

Horizontal Partitioning by Predicate Abstraction and Its Application to Data Warehouse Design

Aleksandar Dimovski[1], Goran Velinov[2], and Dragan Sahpaski[2]

[1] Faculty of Information-Communication Technologies, FON University, Skopje, 1000, Republic of Macedonia
[2] Institute of Informatics, Faculty of Sciences and Mathematics, Ss. Cyril and Methodius University, Skopje, 1000, Republic of Macedonia

Abstract. We propose a new method for horizontal partitioning of relations based on predicate abstraction by using a finite set of arbitrary predicates defined over the whole domains of relations. The method is formal and compositional: arbitrary fragments of relations can be partitioned with arbitrary number of predicates. We apply this partitioning to address the problem of finding suitable design for a relational data warehouse modeled using star schemas such that the performance of a given workload is optimized. We use a genetic algorithm to generate an appropriate solution for this optimization problem. The experimental results confirm effectiveness of our approach.

Keywords: Data Warehouse, Horizontal Partitioning, Predicate Abstraction.

1 Introduction

Given a database shared by distributed applications in a network, the performance of queries would be significantly improved by proper data distribution to the physical locations where they are needed most. This can be achieved by using *partitioning* (or *fragmentation*). Partitioning is the process of splitting large relations (tables) into smaller ones so that the DBMS does not need to retrieve as much data at any one time. There are two ways to partition a relation: *horizontally* and *vertically*. Horizontal partitioning involves splitting the tuples (rows) of a relation, placing them into two or more relations with the identical structure. Vertical partitioning involves splitting the attributes (columns) of a relation, placing them into two or more relations linked by the relation's primary key. Advantages that partitioning brings are the following: it can significantly impact the performance of the workload, i.e. set of queries that executes against the database system by reducing the cost of accessing and processing data; it allows parallel processing of data by locating tuples where they are most frequently accessed, etc.

Data Warehouses (DW) store active data of business value for an organization. Relational DW often contain large relations (fact relations or fact tables) and require techniques both for managing these large relations and for providing good query performance across these large relations. Our goal is to find an optimal partitioning scheme of a data warehouse for a given representative workload by using partitioning on relations and indexes. An important issue about partitioning is to which degree should it occur. We need to find a suitable level of partitioning relations within the range starting from single attribute values or tuples to the complete relations. The space of possible physical partitioning scheme alternatives that need to be considered is very large. For example, each relation can be partitioned in many different ways.

In this paper we describe a new formal approach for horizontal partitioning and its application for optimizing data warehouse design in a cost-based manner. Horizontal partitioning is based on predicate abstraction which maps the domain of a relation to be partitioned to an abstract domain following a finite set of arbitrary predicates chosen over the whole concrete domain. To address the above optimization problem, we first choose a set of predicates to horizontally partition some (or all) dimension relations of a DW with star scheme, and then split the fact relation by using the predicates specified on dimension relations. This creates a number of sub-star fragments of the data warehouse we consider, where each sub-star fragment consists of a partition of the fact table and corresponding to it partitions of dimension relations. Then we use a genetic algorithm, known evolutionary heuristic, to find a suitable solution which minimizes the query cost. Our method does not guarantee an optimal partitioning, but the experimental results suggest that it produces good solutions in practice.

Organization. After discussing related work, in Section 2 we formally present a procedure for horizontal partitioning of relations based on predicate abstraction. Section 3 contains brief review of relational DW with star scheme model. In Section 4, the optimization problem is defined and a genetic algorithm addressing it is described. We present experimental results in Section 5. Finally, in Section 6, we conclude and discuss future work.

Related Work. The work on optimal partitioning of a database design for a given representative workload and its allocation to a number of processor nodes has been extensive [13,19]. The ideas were then adapted to the setting of a data warehouse [1,6]. In [7] is proposed a technique for materializing data warehouse views in vertical fragments, aimed to tightly fit the reference workload. In our previous works, we develop a technique for optimizing a data warehouse scheme by using vertical partitioning [16,17], and then extend it by defining multiversion implementation scheme in order to take into account the dynamic aspect of a warehouse due to the changes of the scheme structure and queries [15].

Predicate abstraction (or boolean abstraction) has been widely used in model checking [8]. The idea of predicate abstraction is to map concrete states of a system to abstract states according to their evaluation under a finite set of predicates. Automatic predicate abstraction has been developed for verifying infinite-state systems such as software programs [2,5].

Horizontal partitioning has also been used for optimizing the performance of queries [4,12,14]. However, the method has not been formalized before, and in such way, it has not been applied in a concrete algorithm. The work presented in this paper is close to [3], which also uses horizontal partitioning for selecting an optimal scheme of a warehouse. However, our work brings several benefits.

- We formally define the method of horizontal partitioning of a relation by using the notion of predicate abstraction.
- In our optimizing procedure we use arbitrary predicates which can be defined over the whole domain of a relation, not as in [3] where only atomic predicates applied to single attributes are used.
- Our partitioning method is compositional, which enables partitioning arbitrary fragments of relations with arbitrary number of predicates.
- A global index table is created which maintains pointers to each of the sub-star fragments.

In our experiments we use genetic algorithms that are also used for optimization of a data warehouse schema in [3] and [18]. We conducted the experiments using the Java Genetic Algorithm Framework JGAP [11].

2 Horizontal Partitioning by Predicate Abstraction

Let R be a relation, and $A_1, ..., A_n$ be its attributes with the corresponding domains $Dom(A_1), ..., Dom(A_n)$. A *predicate* represents a pure boolean expression over the attributes of a relation R and constants of the attributes' domains. An *atomic predicate* p is a relationship among attributes and constants of a relation. For example, $(A_1 < A_2)$ and $(A_3 >= 5)$ are atomic predicates. Then, the set of all predicates over a relation R is:

$$\phi ::= p \,|\, \neg\phi \,|\, \phi_1 \wedge \phi_2 \,|\, \phi_1 \vee \phi_2$$

We define *horizontal partitioning* as a pair (R, ϕ), where R is a relation and ϕ is a predicate, which partitions R into at most 2 fragments (sub-relations) with the identical structure (i.e. the same set of attributes), one per each truth value of ϕ. The first fragment includes all tuples t of R which satisfy ϕ, i.e. $t \vDash \phi$. The second fragment includes all tuples t of R which do not satisfy ϕ, i.e. $t \nvDash \phi$. It is possible one of the fragments to be empty if all tuples of R either satisfy or do not satisfy ϕ. Note that, the partitioning (R, ϕ) is identical to $(R, \neg\,\phi)$. If we apply the predicate $true$ (or $false$) to a relation, then it remains undivided.

Example 1. Let $R = (A_1\, int, A_2\, int, A_3\, date)$ be a relation. It can be divided into 2 partitions by using one of the following predicates:

- $\phi = (A_1 = A_2)$, which results into a fragment where the values of A_1 and A_2 are equal for all tuples, and a fragment where those values are different.
- $\phi = (A_3 >=' 01 - 01 - 07') \wedge (A_3 <' 01 - 01 - 09')$, which results into a fragment where the values of A_3 are in the range from $'01 - 01 - 07'$ to $'01 - 01 - 09'$, and a fragment where those values are not in the specified range. □

Embedded horizontal partitioning is also allowed. We can apply horizontal partitioning using a predicate ϕ_2 to each of the fragments obtained by a partitioning (R, ϕ_1), denoted as (R, ϕ_1, ϕ_2). In this way, we can split the initial relation R into at most 4 fragments:

$$\begin{aligned} R_1 &= \{t \in R \,|\, t \vDash \phi_1 \ \wedge \ \phi_2\} \\ R_2 &= \{t \in R \,|\, t \vDash \phi_1 \ \wedge \neg \, \phi_2\} \\ R_3 &= \{t \in R \,|\, t \vDash \neg \, \phi_1 \ \wedge \ \phi_2\} \\ R_4 &= \{t \in R \,|\, t \vDash \neg \, \phi_1 \ \wedge \neg \, \phi_2\} \end{aligned}$$

Embedded horizontal partitioning can go on to an arbitrary depth m, such that in each level an arbitrary predicate is applied to the obtained fragments. Embedded horizontal partitioning of a relation R with depth m is denoted as $(R, \phi_1, \phi_2, ..., \phi_m)$ where $\phi_1, \phi_2, ..., \phi_m$ are arbitrary predicates. The partitioning with depth m splits the initial relation R into at most 2^m fragments.

Example 2. Let we have the relation R from Example 1. $\big(R, (A_3 >=' 01 - 01 - 07') \wedge (A_3 <' 01 - 01 - 09'), A_3 <' 01 - 01 - 07'\big)$ splits R into at most 3 fragments:

– R_1 with tuples satisfying $A_3 <' 01 - 01 - 07'$.
– R_2 with tuples satisfying $'01 - 01 - 07' <= A_3 <' 01 - 01 - 09'$.
– R_3 with tuples satisfying $A_3 >=' 01 - 01 - 09'$.

Note that, the final number of fragments depends on the structure of R. For example, if there are no tuples that satisfy $(A_3 >=' 01 - 01 - 09')$ then the fragment R_3 will be empty. □

2.1 Predicate Abstraction

Given a relation $R = (A_1, ..., A_n)$ and a set of predicates $\mathcal{P} = \{\phi_1, \phi_2, ..., \phi_m\}$, we define the concrete domain of R as:

$$Dom(R) = Dom(A_1) \times Dom(A_2) \times \ldots \times Dom(A_n)$$

and the abstract domain of R with respect to $\mathcal{P}$, denoted as $AbsDom(R)_{\mathcal{P}}$, as the set of bitvectors of length m (one bit per predicate $\phi_i \in \mathcal{P}$, for $i = 1, \ldots, m$):

$$AbsDom(R)_{\mathcal{P}} = \{0, 1\}^m$$

The abstraction function is the mapping from the concrete domain $Dom(R)$ to the abstract domain, assigning a tuple t in R the bitvector representing the Boolean covering of t:

$$\begin{aligned} &\alpha : Dom(R) \rightarrow AbsDom(R)_{\mathcal{P}}, \\ &\quad t = (a_1, \ldots, a_n) \mapsto (v_1, \ldots, v_m),\ t \vDash v_1 \cdot \phi_1 \ \wedge \ldots \wedge \ v_m \cdot \phi_m \end{aligned}$$

where $0 \cdot \phi = \neg\phi$ and $1 \cdot \phi = \phi$. The concretization function is the mapping

$$\begin{aligned} &\gamma : AbsDom(R)_{\mathcal{P}} \rightarrow Dom(R), \\ &\quad (v_1, \ldots, v_m) \mapsto \{t \,|\, t \vDash v_1 \cdot \phi_1 \ \wedge \ldots \wedge \ v_m \cdot \phi_m\} \end{aligned}$$

Given a relation $R = (A_1, ..., A_n)$ and a set of predicates $\mathcal{P} = \{\phi_1, \phi_2, ..., \phi_m\}$, the horizontal partitioning $(R, \phi_1, \ldots, \phi_m)$, or $(R, \mathcal{P})$ for short, splits R into at most 2^m fragments:

$$R_{(v_1,\ldots,v_m)} = \{t \,|\, \alpha(t) = (v_1, \ldots, v_m)\}$$

We form an index table with 2^m entries representing all possible bitvectors of length m:

$$\{(v_1, \ldots, v_m) \,|\, v_i \in \{0, 1\},\, i = 1, \ldots, m\}$$

An index entry $(v_1, \ldots, v_m)$ specifies the tuples of R satisfying the entry value with respect to the set of predicates $\mathcal{P}$. So, each single entry $(v_1, \ldots, v_m)$ from the index table points to exactly one fragment $R_{(v_1,\ldots,v_m)}$. If some fragment is empty, then there will be no pointer to it. Then, local index tables are created on each of the fragments.

Example 3. Let us have the relation R from Examples 1 and 2. The partitioning $\big(R, (A_3 >=' 01-01-07') \wedge (A_3 <' 01-01-09'), A_3 <' 01-01-07'\big)$ splits R into the following fragments: $R_{(0,0)}$ with tuples satisfying $A_3 >=' 01-01-09'$; $R_{(0,1)}$ with tuples satisfying $A_3 <' 01-01-07'$; $R_{(1,0)}$ with tuples satisfying $'01-01-07' <= A_3 <' 01-01-09'$; $R_{(1,1)}$ with tuples satisfying both predicates, which is an empty set. The index table contains 4 entries: $(0, 0)$, $(0, 1)$, $(1, 0)$, and $(1, 1)$ pointing to the corresponding fragments. □

2.2 Predicate Selection

We can obtain a set of predicates $\mathcal{P} = \{\phi_1, ..., \phi_m\}$ applicable to a relation R for horizontal partitioning by extracting them from a set of given (input) queries. The predicates are specified in the selection clause of a query. As we have seen, the number of horizontal fragments is in the worst case exponential in the number of predicates involved. Therefore, it is important to use as few predicates as possible. Given $\mathcal{P}$ and R we want to generate a set of complete and minimal predicates $\mathcal{P}_{comin}$ and then partition R by using $(R, \mathcal{P}_{comin})$. A set of predicates is *complete* if it partitions the relation into a set of mutually disjoint fragments such that the access frequency of all tuples within a fragment is uniform for all queries. A set of predicates is *minimal* if the resulting partitioning is obtained by minimal number of predicates. There might be some redundant predicates in $\mathcal{P}$ for our horizontal partitioning algorithm, which lead to no additional fragments.

Example 4. Let we have the relation R from the previous Examples. Consider predicates $\phi_1 = (A_3 >=' 01-01-07') \wedge (A_3 <' 01-01-09')$ and $\phi_2 = (A_3 <' 01-01-07')$. Then (R, ϕ_1, ϕ_2) splits R into 3 fragments as in Example 2. But if we have predicate $\phi_3 = (A_3 >=' 01-01-09')$, then $(R, \phi_1, \phi_2, \phi_3)$ generates again the same 3 fragments as before. So, ϕ_3 is a redundant predicate. Also the partitionings (R, ϕ_1, ϕ_3) and (R, ϕ_2, ϕ_3) are identical to $(R, \phi_1, \phi_2, \phi_3)$. This means that any one of the predicates ϕ_1, ϕ_2, and ϕ_3 can be eliminated. □

The procedure **ComputeMin** for computing a minimal set of predicates based on a given complete set of predicates and a relation is presented in

```
The procedure computes a set of minimal predicates $\mathcal{P}_{min}$ for a given complete set of predicates $\mathcal{P} = \{\phi_1, ..., \phi_m\}$ and a relation $R$.

1 Let $\mathcal{P} = \{\phi_1, ..., \phi_m\}$ be a set of predicates. Let $i := 1$ and $\mathcal{P}_{min} := \emptyset$.
2 If $i > m$, return $\mathcal{P}_{min}$.
3 If $(\phi_i \in \mathcal{P}_{min})$ or $(\neg\phi_i \in \mathcal{P}_{min})$, then $\phi_i$ is redundant. Set $i := i + 1$, and repeat
    from 2.
4 Otherwise, if $\phi_i$ is relevant to $\mathcal{P}_{min}$, then $\mathcal{P}_{min} := \mathcal{P}_{min} \cup \{\phi_i\}$. If there exists a
    $\phi \in \mathcal{P}_{min}$ that is not relevant to $\mathcal{P}_{min} \setminus \{\phi\}$, then set $\mathcal{P}_{min} = \mathcal{P}_{min} \setminus \{\phi\}$. Set
    $i := i + 1$, and repeat from 2.
5 If $\phi_i$ is not relevant to $\mathcal{P}_{min}$, then $\phi_i$ is redundant. Set $i := i + 1$, and repeat
    from 2.
```

Fig. 1. ComputeMin procedure

Figure 1. It checks for each of the predicates whether it can be eliminated or not. We say that a predicate ϕ is *relevant* to a set of predicates $\mathcal{P}$ if there are two tuples t and t' of a fragment F, where $F \in (R, \mathcal{P})$, such that $t \vDash \phi$ and $t' \nvDash \phi$. Note that, it is still possible that some fragments produced from $(R, \mathcal{P}_{min})$ to be empty.

2.3 Derived Horizontal Partitioning

Derived Horizontal Partitioning is defined on a relation table which refers to another relation by using its primary key as reference. Since this relationship will be used during execution of join operations over the two relations, it is of advantage to propagate a horizontal fragmentation obtained for one relation to the other relation and to keep the corresponding fragments at the same place.

Let $R = (A_1, \ldots, A_n)$ and $S = (B_1, \ldots, B_m)$ be relations, A_j $(1 \leq j \leq n)$ be a primary key of R, and B_i $(1 \leq i \leq m)$ be a foreign key of S referring to A_j. Given a horizontal fragmentation of R into $R_1, \ldots, R_k$, then this induces the derived horizontal fragmentation of S into k fragments:

$$S_l = S \ltimes R_l,\ l = 1, \ldots, k$$

where the semi-join operator $\ltimes$ is defined as $S \ltimes R = \pi_{B_1,\ldots,B_m}(S \bowtie R)$, i.e. the result is the set of all tuples in S for which there is a tuple in R that is equal on their common attributes.

3 Data Warehouse Schema

In the core of any data warehouse is a concept of a multidimensional data cube. The data in the cube is stored in specialized relations, called *fact* and *dimension* relations. Fact relations contain basic facts about a model, and they are referencing any number of dimension relations. On the other hand, dimension relations contain extra information about the facts. There are two schemes of implementation: *star* and *snowflake* scheme. In the star scheme all attributes of each dimension are stored in one relation, while in the snowflake scheme attributes in each dimension are normalized and stored in different relations. In this paper we consider data warehouse with star scheme.

Let $(F, D_1, D_2, \ldots, D_k)$ be a star scheme. Given a set $\mathcal{D} = \{D_1, D_2, \ldots, D_k\}$ of dimension relations, let us suppose that each of them D_i $(1 \leq i \leq k)$ is horizontally partitioned by using a set of predicates $\mathcal{P}_i$ into n_i fragments. Then, a fact relation F is partitioned using derived horizontal partitioning in the following way:

$$F_j = \big(...(F \ltimes D_{1r_1}) \ltimes \ldots \ltimes D_{kr_k}\big)$$

where $1 \leq r_i \leq n_i$, $1 \leq i \leq k$, and $1 \leq j \leq \prod_{i=1}^{k} n_i$. So the fact relation will be partitioned into $\prod_{i=1}^{k} n_i$ fragments. Given a fact relation partition $F_j = \big(...(F \ltimes D_{1r_1}) \ltimes \ldots \ltimes D_{kr_k}\big)$, we can create a sub-star scheme fragment $(F_j, D_{1r_1}, \ldots, D_{kr_k})$. If each dimension D_i is partitioned into n_i $(1 \leq i \leq k)$ fragments, then there will be $\prod_{i=1}^{k} n_i$ sub-star schemes in the implementation scheme of a data warehouse.

More formally, if dimension relations are partitioned by sets of predicates $\mathcal{P}_i = \{\phi_{i,1}, \phi_{i,2}, \ldots \phi_{i,m_i}\}$ for $1 \leq i \leq k$, then each dimension relation D_i will be divided into at most 2^{m_i} fragments, and the fact table into at most $2^{\sum_{i=1}^{k} m_i}$ fragments. We can form a global index table with $2^{\sum_{i=1}^{k} m_i}$ entries representing all possible bitvectors of length $\sum_{i=1}^{k} m_i$. An index entry $(v_{1,1}, \ldots, v_{1,m_1}, \ldots, v_{k,m_k})$ specifies the tuples of dimension relations satisfying the entry value with respect to the corresponding set of predicates. Each single entry $(v_{1,1}, \ldots, v_{1,m_1}, \ldots, v_{k,m_k})$ from the index points to exactly one sub-star scheme created by dimension relations $D_{i_{(v_{i,1}, \ldots, v_{i,m_i})}}$ for $1 \leq i \leq k$, and a fact sub-relation $\big(...(F \ltimes D_{1_{(v_{1,1}, \ldots, v_{1,m_1})}}) \ltimes \ldots \ltimes D_{k_{(v_{k,1}, \ldots, v_{k,m_k})}}\big)$. Then, local index tables are created on each of the sub-star schemes.

4 Optimization Problem

As we have seen the number of generated sub-star schemes grows rapidly as the number of fragments of dimensions increases. Thus, it will be difficult for the data warehouse administrator (DWA) to maintain all these sub-star schemes. We want to compute an (near) optimal number of fragments such that the performance of queries will be good and the cost of maintaining and managing so many fragments will be acceptable. The latter is addressed by allowing to choose in our procedure a maximal number of sub-star scheme fragments that DWA can maintain. We now formally define the problem of finding an optimal partitioning implementation scheme of a data warehouse.

4.1 The Optimization Problem

Let $(F, D_1, D_2, \ldots, D_k)$ be a star scheme, $\mathcal{Q} = \{Q_1, Q_2, \ldots, Q_l\}$ be a set of queries, and Cost be a cost evaluation function. The optimization problem is defined as follows. Find a set of sub-star fragments $\mathcal{S} = \{S_1, S_2, \dots, S_N\}$ such that the cost

$$\mathsf{Cost}(\mathcal{S}, \mathcal{Q}) \text{ is minimal}$$

subject to the constraint $N \leq W$, where W is a threshold representing a maximal number of fragments that can be generated. The cost evaluation function is defined according to the linear cost model [9]. The cost of answering a query Q_i, denoted as $\mathsf{Cost}(\mathcal{S}, Q_i)$, is taken to be equal to the space ocupied by the fragment $S_j \in S$ from which the query is answered, i.e. proportional to the total number of rows of the fragment S_j.

4.2 The Optimization Procedure

We now describe an optimization procedure for obtaining an optimal partitioning implementation scheme given a workload:

1 Extract all predicates $\mathcal{P}$ used by $\mathcal{Q}$.
2 Find a complete set of predicates $\mathcal{P}_i \subseteq \mathcal{P}$ $(1 \leq i \leq k)$ corresponding to each dimension relation D_i.
3 Use **ComputeMin**$(\mathcal{P}_i, \mathcal{D}_i)$ procedure to find a minimal set of predicates for each relation.
4 Apply a genetic algorithm to find an optimal partitioning scheme.

Genetic algorithm (GA) [10] is a search method for finding approximate solutions to optimization problems. It uses techniques inspired by evolutionary biology such as mutation, selection, crossover, and survival of the fittest. Candidate solutions to a given problem, also called *chromosomes*, are represented most commonly as bit strings, but other encodings are also possible. The algorithm starts from a population of randomly generated solutions and happens in iterations (i.e. generations). In each generation, the cost of every solution in the population is evaluated, multiple solutions are selected from the current population based on their cost, and modified (recombined and possibly randomly mutated) to form a new population. The new population is then used in the next iteration. The algorithm terminates when either a maximum number of generations has been produced, or a solution with satisfactory cost has been found. We now present the design of our genetic algorithm.

Representation of Solution. Let $\mathcal{P}_i = \{\phi_{i,1}, \phi_{i,2}, \dots \phi_{i,m_i}\}$ $(1 \leq i \leq k)$ be a complete and minimal set of predicates that needs to be applied to the dimension D_i for horizontal partitioning. A possible solution of our problem is a set of N $(N \leq W)$ different sub-star fragments. Each fragment S_j $(1 \leq j \leq N)$ is represented by a bit array (or, bit-vector).

$$(v_{1,1}, \dots, v_{1,m_1}, \dots, v_{k,1}, \dots, v_{k,m_k})$$

containing one bit for each predicate used in the partitioning. Each bit in the solution is set to 1, if the respective predicate is satisfied by all tuples in S_j; otherwise it is set to 0. So, we have that

$$S_j = F_{(v_{1,1},\dots,v_{1,m_1},\dots,v_{k,m_k})} = \{t \mid \alpha(t) = (v_{1,1},\dots,v_{1,m_1},\dots,v_{k,m_k})\}$$

or

$$S_j = F_{(v_{1,1},\dots,v_{1,m_1},\dots,v_{k,m_k})} = \Big(...(F \ltimes D_{1_{(v_{1,1},\dots,v_{1,m_1})}}) \ltimes \dots \ltimes D_{k_{(v_{k,1},\dots,v_{k,m_k})}}\Big)$$

The entry from the local index table which points to S_j will be its bit array representation $(v_{1,1},\dots,v_{1,m_1},\dots,v_{k,1},\dots,v_{k,m_k})$. In this way, we obtain that the search space of our optimization problem is $2^{N\sum_{i=1}^{k} m_i}$, or in the worst case it is $2^{W\sum_{i=1}^{k} m_i}$.

A chromosome consists of N composite genes, where each composite gene is a bit-vector representing one fragment S_j as described above. One chromosome represents one possible solution to the problem.

Genetic Algorithm Operators. A single point crossover operator is used, which chooses a random bit from two parent chromosomes, i.e. solutions, and then performs a swap of that bit and all subsequent bits between the two parent chromosomes, in order to obtain two new offspring chromosomes.

The mutation operation is performed over each gene of a chromosome and mutates them with a given probability. Because the genes are represented as bit arrays, a mutation of a gene means fliping the value of every bit with the given probability.

We use a natural selection operator where a chromosome is selected for survival in the next generation with a probability inversely proportional to the cost of the solution represented by the chromosome. A strategy of elitist selection is also used where the best chromosome of the population in the current generation is always carried unaltered to the population in the next generation.

The termination of the GA is established as follows. We perform a number of GA experiments, and we determine the number of iterations that are needed for the GA, such that no significant improvement in the solution quality can be detected for a specified number of iterations.

5 Experimental Results

The experiments were performed by using four sets of 25, 50, 100 and 200 distinct queries on a star scheme with 4 dimensions, which contain 11 attributes, and 1 fact table with size of $1.25 * 10^9$ rows. Each query contains a selection clause of the form: $\phi_1 \wedge ... \wedge \phi_n$. The sets of 25, 50, 100 and 200 queries are composed of 178, 341, 711 and 1387 predicates, respectively. The number of predicates in a query is generated using a gaussian distribution with mean 7 and standard deviation 1. The attribute and its value in a given predicate are generated using

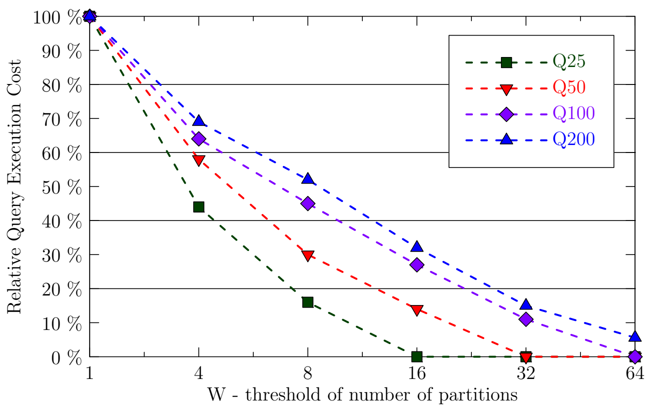


Fig. 2. The query cost for different values of the threshold W

a uniform distribution on the set of attributes and the domain of the selected attribute, respectively. The termination condition of all the experiments is set to 200 iterations and the population size is set to 200 chromosomes.

In Figure 2, we show the query execution cost for different query sets and different values of the threshold W. The query execution cost is represented in percentage relative to the worst query execution cost (on a star scheme with no partitioning) for the given query set. We can see that the query execution cost on a partitioned star schema is reduced by the order of 10^3 compared to an unpartitioned schema. Also, note that the query cost reduces when the threshold increases.

In Figure 3, we compare the relative query execution costs on three query sets composed of 100 queries each, which contain predicates generated using a gaussian distribution with mean 3, 7 and 15 and standard deviation 1, on the

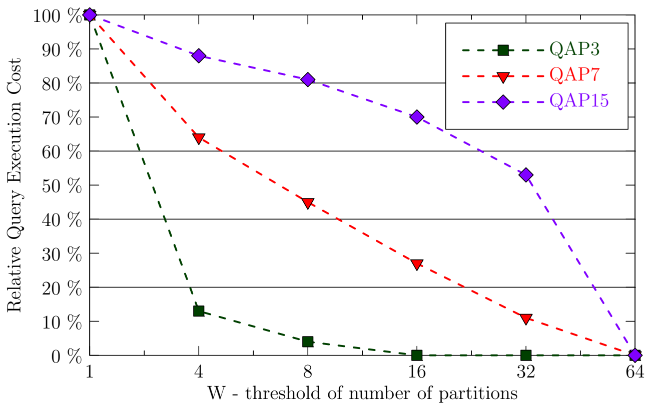


Fig. 3. The cost of query sets with different average number of predicates

same star schema used in Figure 2. The three sets of queries with 3, 7 and 15 average number of predicates per query are composed of 305, 711 and 1100 predicates, respectively. It can be seen that the query execution cost reduces more rapidly when the average number of predicates in the generated queries is smaller.

6 Conclusion

In this paper we present a novel formal approach for horizontal partitioning of relations based on predicate abstraction. Then, we show how to use this approach for finding an optimal data warehouse design which takes account of the performance of queries and the maintenance cost.

A possible direction for extension is to combine our partitioning method with vertical partitioning, and see its effects on the problem of computing an optimal data warehouse design. It is also interesting to extend the proposed approach to dynamically evolving data warehouse, which can change its scheme structures and its queries.

References

1. Agrawal, S., Narasayya, V., Yang, B.: Integrating Vertical and Horizontal Partitoning into Automated Physical Database Design. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 359–370 (2004)
2. Ball, T., Podelski, A., Rajamani, S.K.: Boolean and Cartesian Abstraction for Model Checking C Programs. In: Margaria, T., Yi, W. (eds.) TACAS 2001. LNCS, vol. 2031, p. 268. Springer, Heidelberg (2001)
3. Bellatreche, L., Boukhalfa, K.: An Evolutionary Approach to Schema Partitioning Selection in a Data Warehouse. In: Tjoa, A.M., Trujillo, J. (eds.) DaWaK 2005. LNCS, vol. 3589, pp. 115–125. Springer, Heidelberg (2005)
4. Bellatreche, L., Karlapalem, K., Simonet, A.: Algorithms and Support for Horizontal Class Partitioning in Object-Oriented Databases. The Distributed and Parallel Databases Journal 8(2), 155–179 (2000)
5. Dimovski, A., Ghica, D.R., Lazić, R.: Data-Abstraction Refinement: A Game Semantic Approach. In: Hankin, C., Siveroni, I. (eds.) SAS 2005. LNCS, vol. 3672, pp. 102–117. Springer, Heidelberg (2005)
6. Furtado, P.: Experimental Evidence on Partitioning in Parallel Data Warehouses. In: Proceedings of the 7th ACM International Workshop on Data Warehousing and OLAP (DOLAP), pp. 23–30 (2004)
7. Golfarelli, M., Maniezzo, V., Rizzi, S.: Materialization of Fragmented Views in Multidimensional Databases. Data & Knowledge Engineering 49(3), 325–351 (2004)
8. Graf, S., Saidi, H.: Construction of Abstract Atate Graphs with PVS. In: Grumberg, O. (ed.) CAV 1997. LNCS, vol. 1254, pp. 72–83. Springer, Heidelberg (1997)
9. Harinarayan, V., Rajaraman, A., Ullman, J.D.: Implementing data cubes efficiently. In: Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, vol. 25(2), pp. 205–216. ACM Press SIGMOD Record, New York (1996)

10. Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press (1995)
11. Meffert, K.: JGAP - Java Genetic Algorithms and Genetic Programming Package, http://jgap.sf.net
12. Ozsu, M.T., Valduriez, P.: Principles of Distributed Database Systems. Prentice-Hall, Englewood Cliffs (1999)
13. Sacca, D., Wiederhold, G.: Database Partitioning in a Cluster of Processors. Proceedings of the ACM Transactions on Database Systems (TODS) 10(1), 29–56 (1985)
14. Sanjay, A., Narasayya, V.R., Yang, V.R.: Integrating Vertical and Horizontal Partitioning into Automated Physical Database Design. In: Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data, pp. 359–370. ACM Press SIGMOD Record, New York (2004)
15. Sahpaski, D., Velinov, G., Jakimovski, B., Kon-Popovska, M.: Dynamic Evolution and Improvement of Data Warehouse Design. In: Proceedings of Balkan Conference in Informatics, IEEE Computer Society's Conference Publishing (IEEE BCI), pp. 115–125 (2009)
16. Velinov, G., Gligoroski, D., Kon-Popovska, M.: Hybrid Greedy and Genetic Algorithms for Optimization of Relational Data Warehouses. In: Proceedings of Multi-Conference: Artificial Intelligence and Applications (IASTED), pp. 470–475 (2007)
17. Velinov, G., Jakimovski, B., Cerepnalkoski, D., Kon-Popovska, M.: Framework for Improvement of Data Warehouse Optimization Process by Workflow Gridification. In: Atzeni, P., Caplinskas, A., Jaakkola, H. (eds.) ADBIS 2008. LNCS, vol. 5207, pp. 295–304. Springer, Heidelberg (2008)
18. Yu, J.X., Yao, X., Choi, C., Gou, G.: Materialized Views Selection as Constrained Evolutionary Optimization. Proceedings of IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews 33(4), 458–468 (2003)
19. Zilio, D.: Physical Database Design Decision Algorithms and Concurrent Reorganization for Parallel Database Systems. Ph. D. Thesis, University of Toronto (1998)

Checkable Graphical Business Process Representation

Sven Feja[1], Andreas Speck[1], Sören Witt[1], and Marcel Schulz[2]

[1] Christian-Albrechts-University Kiel
Olshausenstrasse 40, 24098 Kiel, Germany
[2] Intershop Communications AG,
Intershop Tower, 07740 Jena, Germany

Abstract. There are different model types to model business processes, like ARIS models, BPMN or UML activity diagrams. These models are well elaborated. Moreover, almost all commercial systems or web-based systems are driven by their dynamic behavior which needs to be described precisely by the business process models. The challenge is the validation of these business process models against behavioral dynamic rules. However, the question is what is to be checked in detail and how this is represented in the models and how the results of the checks are displayed.

In the paper we present a graphical representation supporting the checking business process models. A graphical specification of business rules and regulations is presented which allows to display both the business process models and the rules in one graphical editor. Both models are transformed into a formal language which may be processed by a verification tool – a model checker in our case. The graphical representation is realized with Eclipse which allows to integrate different other verification systems and to extend the current implementation.

Keywords: business process models, process verification, graphical representation of process models, graphical rules representation.

1 Introduction

There are many different model types for business processes. This stresses the importance of processes in commercial systems. And when business processes are that crucial the models of the processes and the correctness of these models are of high importance.

In the paper we focus on business systems and model types like Event-Process Chains (EPCs) [21]. Instead of deriving new (formal) models from these for verification purposes, we propose to integrate the means which are necessary for verification into the existing modeling environment. The goal is to provide an easy-to-understand checking solution for the domain engineer who is usually not a formal methods expert.

The commercial application we use to demonstrate our approach is a high-performance e-commerce system: Intershop Enfinity (Intershop Communications

AG). This e-commerce system is used in large scale systems of retailers (Otto or Quelle), in the automotive branch (Volkswagen, MAN, BMW) or in e-procurement systems (run by the German Federal Ministry of the Interior or governments of other countries or large companies).

The in the following we present an overview over the typical model type in the e-commerce (ARIS EPC). Section 2 outlines related work in the domain of business process verification. Section 3 describes our *Temporal Logics Visualization Framework* (TLVF) with its elements (in particular its visual temporal logic language and its visual error representation) followed by a demonstrating example and a conclusion. In section 4 the modeling tool (BAM – *Business Application Modeler*) is presented.

1.1 ARIS for Enfinity Modeling Approach

ARIS (*ARIS* (**Ar**chitecture of integrated **I**nformation **S**ystems) is a well-known approach supporting business systems in general. *ARIS for Enfinity* is a specific profile for the modeling of Intershop Enfinity e-commerce systems [3].

The Enfinity-based e-commerce systems are modeled in various model types. The model type used is mainly the EPC. Further models are the value added chain or function hierarchies.

The EPCs are used to model the business processes on a specific detail level (cf. model elements description in section 1.2). EPCs are more concrete than value added chains and present the business aspects of the processes very well. However, they are no concrete implementation models e.g. like UML sequence charts (here *ARIS for Enfinity* provides *Pipeline Models* which are executed by an application server). The EPC models are ideal for the communication between the domain experts (economists) and the computer scientists, since they are still understood by both groups.

Based on the EPC models the design of the implementation may start. In the case of Intershop Enfinity, executable workflow models (called *Pipelines*) represent the design and are executed by the system's application server.

When the domain experts want to check the business process descriptions of an e-commerce system, this is generally performed on the level of EPC models. Therefore, business rules, regulations and system specific requirements which have to be implemented by the system are to be verified on this business process (EPC) level. If the EPC models do not represent the required rules and regulations correctly then the resulting system will hardly meet the needs. Therefore it is essential to verify the EPC models.

1.2 EPC Model Elements

The EPC model type is part of the modeling concept. For research purpose we use a prototype modeling tool based on *ARIS* and *ARIS for Enfinity* which has been developed in cooperation with the tow originators IDS Scheer and Intershop. This system *Temporal Logics Visualization Framework* (TLVF) is presented in subsection 3.1.

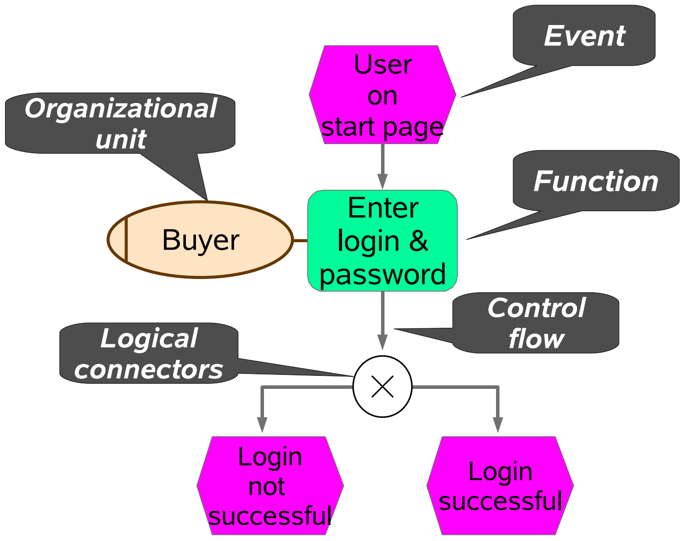


Fig. 1. EPC Model

The basic elements of an EPC model are shown in figure 1 (in the middle): The control flow is symbolized by a sequence of events (magenta hexagons) and functions (green rectangles with rounded edges) which are connected by arrows representing the control flow. Branches in the control flow are defined by the Boolean logic operators: AND, OR and XOR. In the figure an XOR is depicted. AND requires that all paths of the branch are active. OR indicates that at least one path is used. XOR allows that one and only one path is chosen.

2 Related Work

In the paper we present the integration of means to verify business process models directly into these models. This reduces the gap between domain knowledge and formal methods expertise. Model as well as specification are on the same abstraction level. One part of related work may be found in the field of requirements specification. A comparison of rule elicitation and specification/definition methodologies can be found in [17],[9]. According to [9], the use of formal and automatic methods is "the least ambiguous regulations and rules representation allowing for automatic verification techniques." A push-button formal technique to verify the fulfillment of automata-based specifications is model checking [4]. It uses (temporal) logics as specification language and checks these (temporal) rules against a model. As opposed to other formal approaches (e.g. theorem provers) it is more restricted in what can be verified leading, on the other hand, to the advantage of a developer-friendly extensive automation.

A model used model checkers are often finite state machines (e. g. [4]). The checking of software systems in general has been quite early. The early approaches motivate our work and demonstrate the general applicability of model checking for software systems.

The usage of model checking requires the input of formal specification formulas/rules and finite state machine models. Therefore, any input needs to be transformed to such a formal format. The core idea of our approach is similar to the idea of Model-Driven software development [22]. High-level platform independent models are successively transformed to lower-level platform dependent models. Some examples for a powerful transformation concept which is base of our transformation are [12] [13]. An approach demonstrating the transformation from higher abstraction levels to formal models for the purpose of verification is [8]. Other approaches transform business process models not directly into checkable models but first to transform them into Petri-Net models and then to check the Petri-Net models. [7] describe such a procedure.

The rules and regulations the business process models have to fulfill are expressed on the same (higher, visual) abstraction level as these business process models. The path to raise the abstraction level of models, rules and error representation may also be found in [5] and [15], for instance.

Other interesting approaches to raise the abstraction level of the rules may be found in mapping natural language to formal descriptions as well as in visualizing business processes and rule models. For instance, [6] provides temporal logic patterns to ease the mapping of natural language to temporal formulas.

A very similar approach to ours is [11]. Here UML activity diagrams are used to model business processes and then checked by an LTL-based model checker. The focus here is the direct transformation to the checker and then the presentation of the result of the model checker. The focus here is on this LTL checking and to explore the usability of this kind of checking concept. This makes clear that not only EPC models are useful for the modeling of business processes but also UML (especially activity diagrams) and BPMN as well [16]. These modeling languages are issue of our future work.

Besides model checking there are other promising verification approaches. One is constraint solving. [20] presents such an approach. The modeling concept we present in this paper is intended to integrate such approaches as well in future.

3 Business Process Modeling

In our approach we combine the graphical business process models with the graphical representation of the (temporal) logic rules. This makes it possible that the rules for processes are on the same level of abstraction as the business process models. Given specification properties which are typically difficult to understand and available in a textual format, we have to deal with the challenge to provide graphical models for them. We solve this task by means of the Temporal Logics Visualization Framework (TLVF [10]). Subsection 3.1 gives a short overview of the Framework. Subsection 3.2 provides our approach to handle the errors detected by the checker.

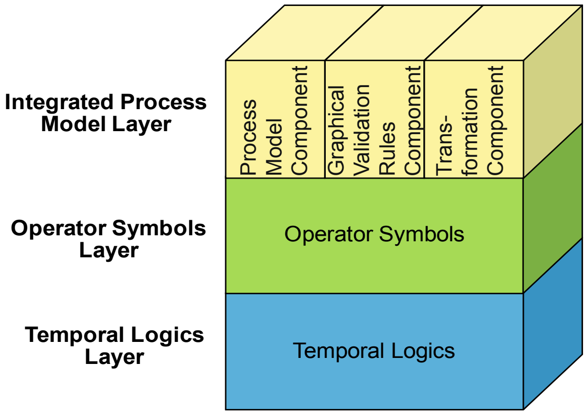


Fig. 2. Temporal Logics Visualization Framework

3.1 Temporal Logics Visualization Framework – TLVF

TLVF is the organizational framework of our visualization concept. It supports all tasks for the visualization of temporal logics in combination with business process models. As depicted in figure 2 the framework is divided in three layers. The first aggregates the logics. The second defines the symbols for each operator of a logic and the third layer delivers the process model, the graphical rules definition and the required transformation tasks.

In general the TLVF allows to state graphical rules of any aggregated logic (e. g. CTL and LTL) in combination with desired process models (e. g. EPC or BPMN).

The general graphical representations of **G**raphical **CTL** (G-CTL) is depicted in figure 3. The graphical logic is defined by their *operator symbols*. We added so called *placeholders*. These may represent functions and events of the EPC model. The operator symbols are the corresponding graphical representation of the textual operator (e. g. quantifiers and Boolean operators). To define rules with those symbols a connection with the process model elements is required. Therefore, the placeholder can be filled with an appropriate process model element. The dashed rectangles (a and b) are the placeholders.

G-CTL operators are based on CTL operators [2]. In CTL there are two types of operators which are combined pairwise: Path quantors always (A) and exists (E) which indicate the occurrence within a path. The temporal operators determine the temporal order. The most important temporal operators are: in the future (F), globally (G), next (X) and until (U). Examples for pairwise combinations are: AG always globally or EX exists next.

This visualized logic can be combined with any desired process model. In our case, G-CTL is combined with the process model event-driven process chain (EPC). The name of the notation of rules for EPCs specified in G-CTL is **EPC-G-CTL** which results from the names of the used components.

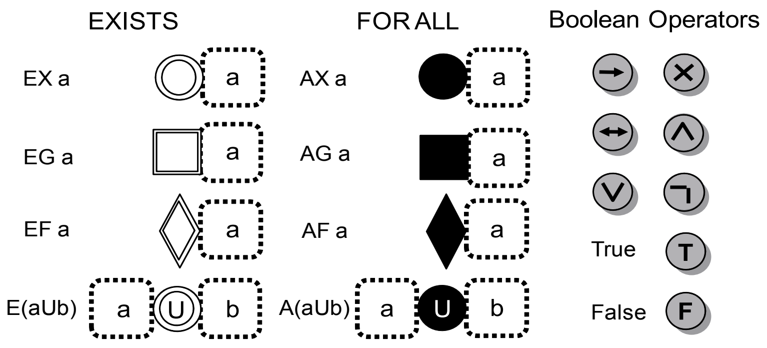


Fig. 3. Symbols of G-CTL

3.2 Validation Errors

Besides the the graphical specification of the rules and regulations as well as the business process models, the systems have to be checked. This is done by a model checker. Two systems are available nuSMV or CoV (Component Verifier) at the moment. Therefore the model and the specification are transformed into a format which can be processed by the checking tool. The concrete process is presented in [19]. The CoV model checker is a symbolic model checker prototype with the goal to reduce the gap between low-level formal models and higher-level component-oriented software systems and without further consideration of performance optimization. Properties to be checked may be assigned to states as well as recursively to properties.

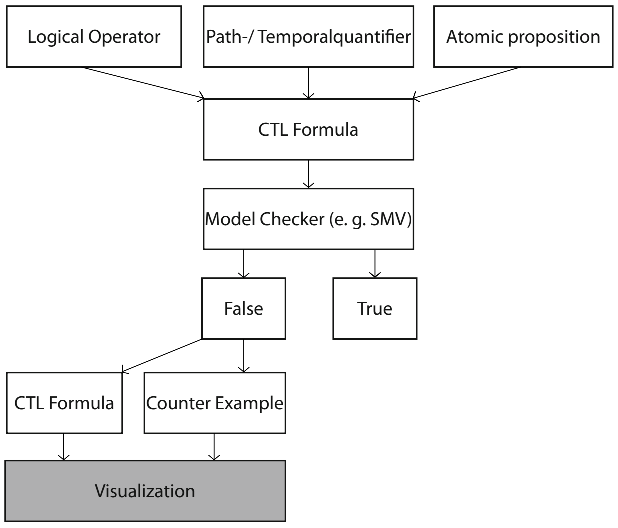


Fig. 4. Visualization of error trace

If the model checker passes with no violated rule nothing has to be visualized. In case of a violated rule there the checker presents a counter-example. This describes a state of error of the model. Next to this counter-example the error causing rule is delivered. Both are the starting point of the visualization process of our approach and are shown on the bottom of figure 4.

When a specific rule is not satisfied the process paths which need to be regarded may be identified. The model checker delivers one or more paths which are the counter-examples.

Currently we are working on the presentation of the error. It is intended to depict the problem graphically. There are different ways to visualize the violation in the process model. One would be the visualization of all witness scenarios in the terms of the original model. This was done for UML in [16]. A second way is the presentation of the counter-example in terms of the original model as done in [14] by generating a sequence diagram of the problem scenario. In contrast to [16] and [14] our approach presents the results of the validation in the original process model. The main advantage is a view on the previously modeled process which directly shows the point of violation. Additionally, this allows it to provide adaption advice for a validation error. This is achieved by an interpretation of the validation error.

4 Process and Rule Modeling

Figure 5 depicts a typical, however still rather small example of a (sub-) business process (named *Order finalization process*). This is a typical model of the Enfinity e-procurement system. The model describes the approval functionality in an e-procurement system modeled as EPC. This model is one small sub-process out of a set of some tens to several hundreds of such sub-processes (the number depends on the complexity of the system), which represent the complete business process description.

When a tool – like TLVF (*Temporal Logics Visualization Framework*, cf. section 3.1) – is used each of these sub-models is developed in a screen page of its own. Other tools similar to TLVF realize the connections in-between these sub-models as hyperlinks.

In the e-commerce system domain there exist business rules (as in most other domains) to which the business process models have to keep. In most cases these rules are based on experience or legal regulations. Some examples for such rules are:

1. An order is always to be completed by the payment and the order confirmation. This seems to be simple. However, when a large number of sub-models are developed there is a certain risk, that there might occasionally exist a path which bypasses the payment. Maybe the payment procedure is bypassed for testing purposes.
2. If the condition that an *Authorization is required* exists then the next step is the *Approval decision.*
3. An order is only valid when the *Purchase information* is *complete* and either an *Authorization is not required* exclusive or an *Authorization is required* and the *Purchase request* is *approved.*

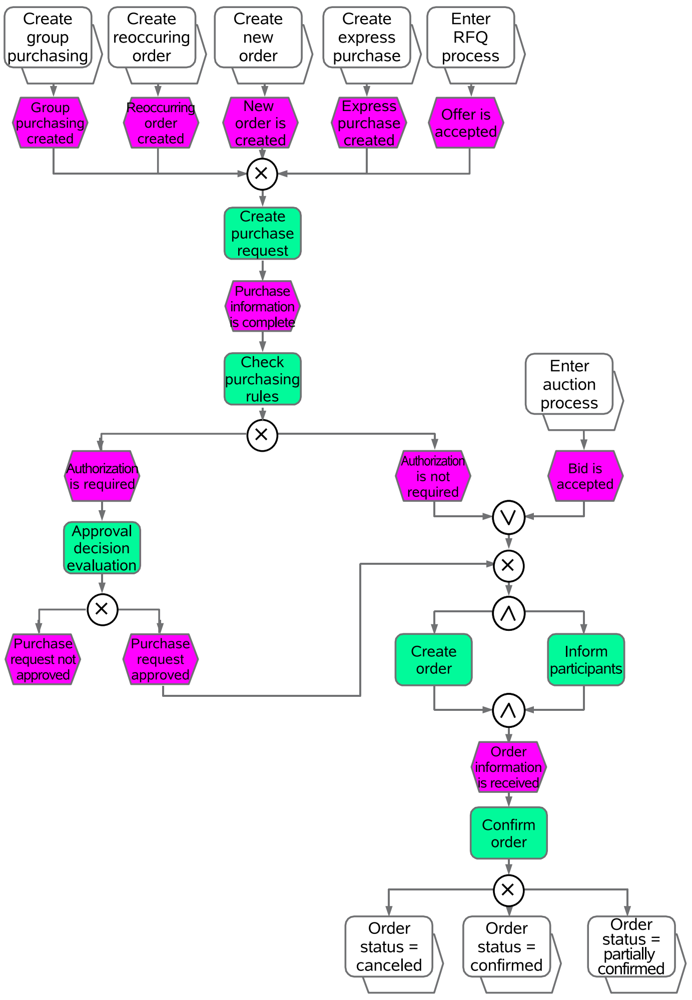


Fig. 5. Example: Approval Procedure in the *Order finalization process* of an e-Procurement System

The first example is a simple check if all paths contain the payment function. This problem may also be detected by a manual check. However, an automated check may save time and is less error-prone. Below in this paper we focus on the verification of the second and third example since they point to some typical problems.

4.1 BAM – Business Application Modeler

The *Business Application Modeler* (BAM) realizes the TLVF modeling concept (cf. subsection 3.1). It supports the business process model (EPC) and the

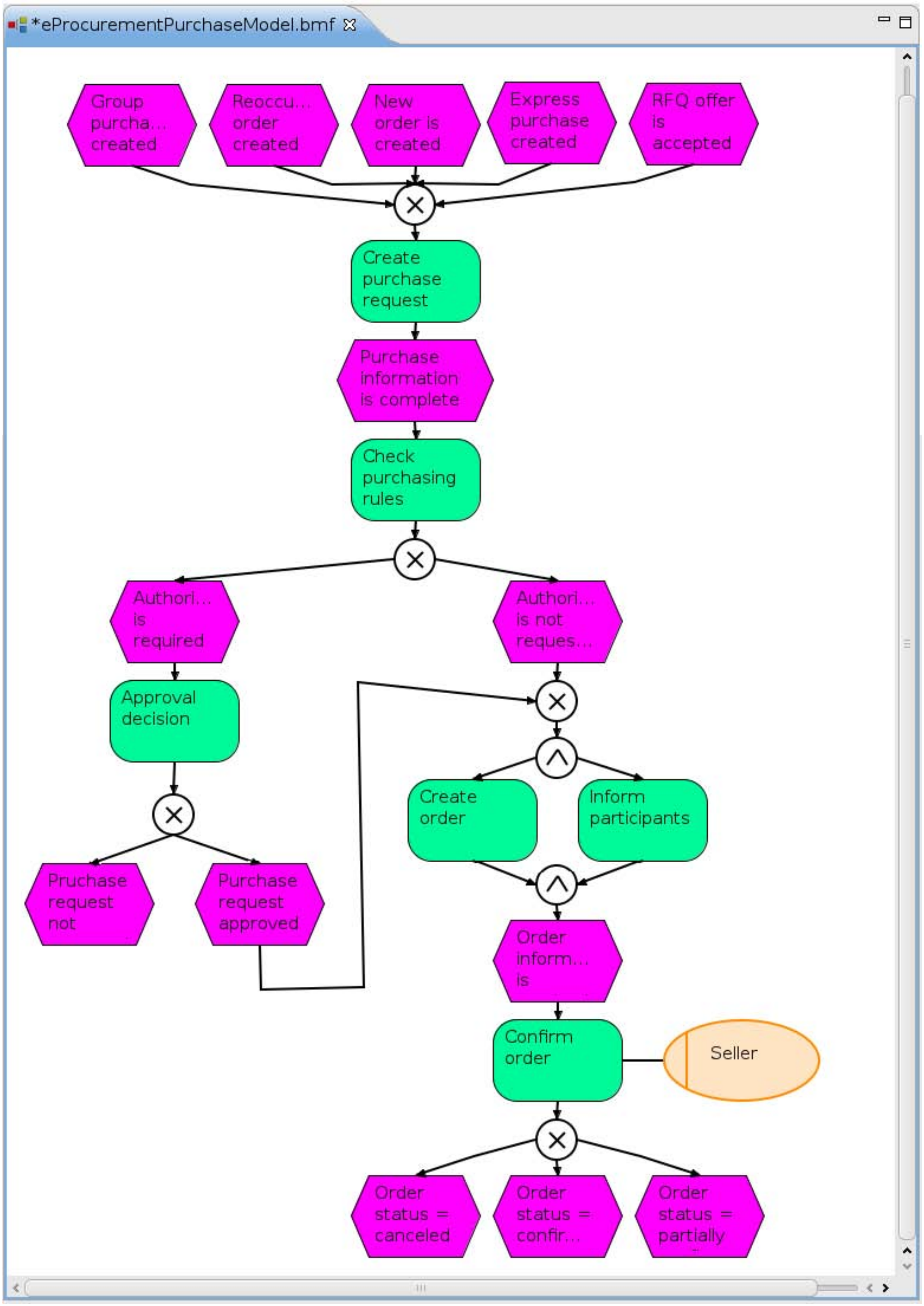


Fig. 6. BAM View of Approval Procedure in the *Order finalization process* of an e-Procurement System

graphical rule notation G-CTL. BAM is based on the Eclipse Graphical Editing Framework (GEF) [1]. GEF supports the required presentation functions and is comparatively portable which means that the BAM editor runs on different operating system platforms. The goal of this Eclipse-based implementation is a high degree of portability and the ability to integrate transformation and checking systems as simple as possible.

Figure 6 depicts the BAM view of the e-procurement model of figure 5. All modeling elements of the ARIS EPCs are available. In this sense there is no difference to any other EPC editor like ARIS or ViFlow.

The EPC models may be sequentialized and the stored as files. Alternatively, the models may be converted to an XML model which is then the base for any transformation into another format, e.g. the format required by a model checker. Currently not realized but desirable would be the possibility to import models from other modeling tool. However, since the technical base is given this will be issue of future work.

4.2 BAM – Rule Editor

Like other EPC editors is supports defining business rules as already discussed. In contrast to these other editors BAM provides a visualization rules. An example of a simple rule is depicted in figure 7. This is actually the already presented

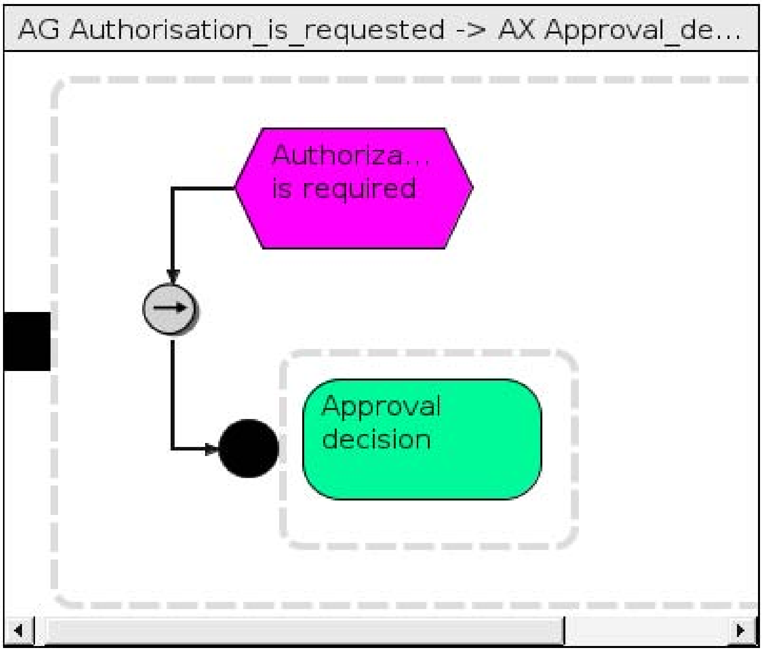


Fig. 7. BAM Rule Visualization

rule (rule 2): If the condition that an *Authorization is required* exists then the next step is the *Approval decision*. The figure shows how the temporal logic is represented graphically. Logical operators (the implication in this case) are represented by symbols which may be connected with other operators (logical or temporal logical operators) and model elements such as events or functions. Temporal logical operators like the AG (always globally) and AX (always next) are realized as containers since they embrace as operands a sub-formula or element.

The CTL formula in the header of the window is just a name for the rule:

$AGAuthorization\_is\_required- > AXApproval\_decision$
Since our development team is experienced in CTL they use the CTL notation to name the rules. However, other users, e.g. domain experts, may name the rules different. It is just the name of the rule and the window representing the rule.

BAM supports the modeling of the rules by providing all required G-CTL operators. Moreover, it presents the set of elements used in the business process model. By this it is impossible to create a rule which does contain element not being in the business process model. If a user wants to create such a rule intentionally it comes clear that this rule is violated and either the rule or the model is wrong. This is an additional error detection mechanism by checking the set of elements of the business process model.

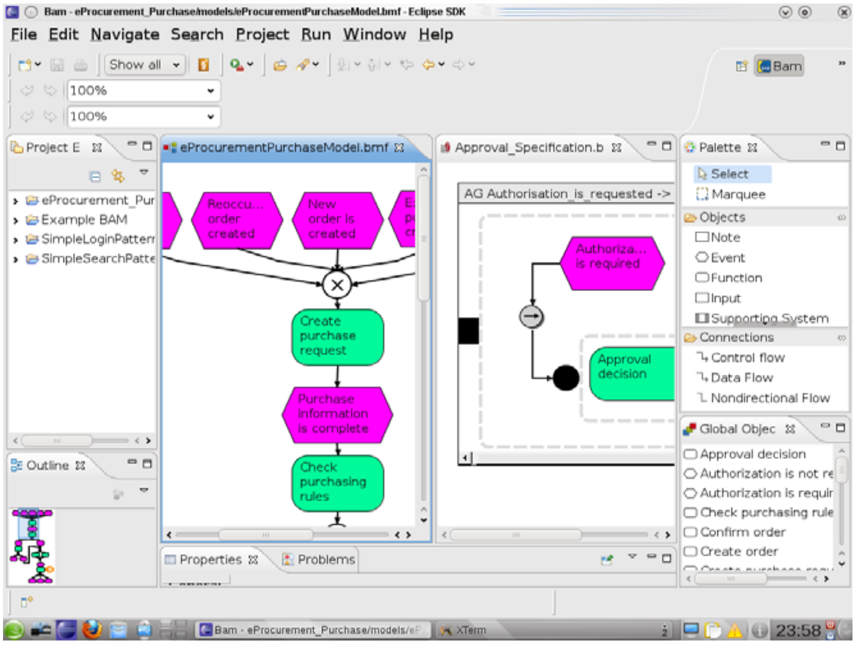


Fig. 8. Screen Shot BAM Editor Overview

The integration of the BAM rule editor is presented in the screen-shot of the complete view of the BAM editor in figure 8. The figure shows a possible composition of the different windows. The model window and the rule window are in the center. The windows supporting the composition of the model and the graphical rule are at the left side. In the bottom window attributes as well as stereotypes used by the elements of the model (and the rule) may be defined.

5 Conclusion and Future Work

The Temporal Logics Visualization Framework (TLVF) enables the visual modeling and validation of temporal rules. In the development of business process based systems (e.g. e-commerce systems) it is essential to assure the correctness of the process models. Since usually the domain experts are not experienced to use checking systems or to formulate rules in a temporal logic language like CTL it is necessary to bridge the gap between the domain experts and the low-level verification viewpoint. The paper proposes a language which allows to express the domain rules like business rules with a similar model type than the business process model types (ARIS EPCs in our case). The graphical logic language G-CTL based on CTL. A modeling tool BAM (*Business Application Modeler*) supports the modeling by realizing the *Temporal Logics Visualization Framework* (TLVF). BAM is implemented with the Eclipse GEF. Eclipse supports the integration of subsystems in an excellent way which is very helpful for the integration of the checking systems.

The BAM visualization concept may be improved by supporting different viewpoints on the models. The domain expert should be able to select different abstraction levels and views on the business process model to reason about rules and regulations. The current BAM version is limited to EPC models. The editor may be extended to support other modeling languages such as BPMN. Therefore, the architecture of BAM allows to display other modeling languages. A generic meta model is of interest to support the integration of different checking systems. Such a generic meta model would be the base of new transformations to other target models than those being processed by model checkers. This means that other checking systems like constraint solvers may be integrated as well.

References

1. Anders, E.: Modellierung und Validierung von Prozessmodellen auf Basis variabler Modellierungsnotationen und Validierungsmethoden als Erweiterung für Eclipse. Diploma Thesis, Christian-Albrechts-University Kiel, Germany (2010)
2. Bérard, B., Bidoit, M., Finkel, A., Laroussinie, F., Petit, A., Petrucci, L., Schnoebelen, P.: Systems and Software Verification – Model-Checking Techniques and Tools. Springer, Berlin (2001)
3. Breitling, M.: Business Consulting, Service Packages & Benefits. Technical report, Intershop Customer Services, Jena, Germany (2002)
4. Clarke, E.M., Grumberg, O., Peled, D.A.: Model Checking, 3rd edn. The MIT Press, Cambridge (2001)

5. Corbett, J.C., Dwyer, M.B., Hatcliff, J., Laubach, S., Păsăreanu, C.S., Robby, Zheng, H.: Bandera: extracting finite-state models from Java source code. In: ICSE 2000: Proceedings of the 22nd International Conference on Software Engineering, pp. 439–448. ACM, New York (2000)
6. Corbett, J.C., Dwyer, M.B., Hatcliff, J., Robby: Expressing Checkable Properties of Dynamic Systems: The Bandera Specification Language. International Journal on Software Tools for Technology Transfer (STTT) 4, 34–56 (2002)
7. De Backer, M., Snoeck, M.: Business Process Verification: a Petri Net Approach. Technical report, Catholic University of Leuven, Belgium (2008)
8. DuVarney, D.C., Purushothaman, I.S.: C Wolf - A Toolset for Extracting Models from C Programs. In: Peled, D.A., Vardi, M.Y. (eds.) FORTE 2002. LNCS, vol. 2529, pp. 260–275. Springer, Heidelberg (2002)
9. Escalona Cuaresma, M.J., Koch, N.: Requirements Engineering for Web Applications A Comparative Study. Journal of Web Engineering 2, 192–212 (2004)
10. Feja, S., Fötsch, D.: Model Checking with Graphical Validation Rules. In: 15th IEEE International Conference on the Engineering of Computer-Based Systems (ECBS 2008), Belfast, NI, GB, pp. 117–125. IEEE Computer Society, Los Alamitos (2008)
11. Förster, A., Engels, G., Schattkowsky, T., Van Der Straeten, R.: Verification of Business Process Quality Constraints Based on Visual Process Patterns. In: Proceedings of the First Joint IEEE/IFIP Symposium on Theoretical Aspects of Software Engineering (TASE 2007), pp. 197–208 (2007)
12. Fötsch, D., Pulvermüller, E.: A Concept and Implementation of Higher-level XML Transformation Languages. Knowledge-Based Systems Journal 22, 186–194 (2009)
13. Fötsch, D., Pulvermüller, E., Rossak, W.: Modeling and Verifying Workflow-based Regulations. In: Proceedings of the Workshop on Regulations Modelling and their Validation & Verification (REMO2V), In Conjunction with the 18th Conference on Advanced Information System Engineering - Trusted Information Systems (CAiSE 2006). Namur University Press (2006)
14. Goldsby, H., Cheng, B.H.C., Konrad, S., Kamdoum, S.: Enabling a Roundtrip Engineering Process for the Modeling and Analysis of Embedded Systems. In: Nierstrasz, O., Whittle, J., Harel, D., Reggio, G. (eds.) MoDELS 2006. LNCS, vol. 4199, pp. 707–721. Springer, Heidelberg (2006)
15. Hatcliff, J., Dwyer, M.B.: Using the Bandera Tool Set to Model-Check Properties of Concurrent Java Software. In: Larsen, K.G., Nielsen, M. (eds.) CONCUR 2001. LNCS, vol. 2154, pp. 39–58. Springer, Heidelberg (2001)
16. Konrad, S., Goldsby, H., Lopez, K., Cheng, B.H.C.: Visualizing Requirements in UML Models. In: REV 2006: Proceedings of the 1st International Workshop on Requirements Engineering Visualization, Washington, DC, USA, vol. 1. IEEE Computer Society, Los Alamitos (2006)
17. Parry, P.W., Özcan, M.B.: The Application of Visualisation to Requirements Engineering (1998)
18. Pulvermüller, E.: Reducing the Gap between Verification Models and Software Development Models. In: Proceedings of the 8th International Conference on New Software Methodologies, Tools, and Techniques (SoMeT 2009), pp. 297–313. IOS Press, Amsterdam (2009)

19. Pulvermüller, E., Feja, S., Speck, A.: Developer-friendly Verification of Process-based Systems. Knowledge Based Systems. Knowledge-Based Systems Journal 23 (to appear 2010)
20. Runte, W.: Modelling and Solving Configuration Problems on Business Processes Using a Multi-Level Constraint Satisfaction Approach. In: The Young Researchers Workshop on Modeling and Management of Business Processes (YRW-MBP 2009). GI LNI, vol. 147, pp. 237–238 (2007)
21. Scheer, A.-W.: ARIS - Modellierungsmethoden, Metamodelle, Awendungen. Springer, Berlin (1998)
22. Völter, M., Stahl, T.: Model-Driven Software Development: Technology, Engineering, Management. John Wiley & Sons, Chichester (June 2006)

Applying the UFO Ontology to Design an Agent-Oriented Engineering Language

Renata S.S. Guizzardi and Giancarlo Guizzardi

Ontology and Conceptual Modeling Research Group (NEMO)
Federal University of Espírito Santo
Av. Fernando Ferrari, S/N, 29060-970, Vitória/ES, Brazil
{rguizzardi,gguizzardi}@inf.ufes.br

Abstract. The problem of designing suitable conceptual modeling languages for system engineering is far from being solved. In the past years, some works have proposed the use of foundational ontologies as analysis tools to enable semantic coherence when (re)designing such languages. In this paper, we exemplify this approach by applying a foundational ontology named UFO in the design of an agent-oriented modeling language for the ARKnowD methodology. Instead of proposing new concepts and constructs, ARKnowD relies on existing work, combining two different approaches, namely Tropos and AORML. Each work is applied in a different development activity, according to their natural propensity: Tropos in Requirements Analysis and AORML in System Design. Besides the ontological approach, we propose some mapping rules between the notations, inspired in the Model Driven Architecture (MDA) metamodel transformation method. This approach helps to guarantee a smooth transition from one activity to the other.

Keywords: conceptual modeling languages design; foundational ontologies, agent-oriented engineering language, model-driven architecture.

1 Introduction

The problem of designing suitable conceptual modeling languages for system engineering is far from being solved. This is a complex matter because, on the one hand, one expects a language which is expressive enough to capture the important aspects of the particular domain in which the system is to be inserted. And on the other hand, one also wants this language to be accessible and provide the right level of abstraction to enable effective analysis and communication with stakeholders. These requirements are often contradictory and hard to achieve.

In the past years, some works have proposed the use of foundational ontologies as analysis tools to enable semantic coherence when (re)designing such languages. Foundational ontologies may be understood as formal systems of domain-independent categories that can be used to characterize the different modes of *existence* and, thus, can be used to characterize the most general aspects of concepts and entities that belong to different domains in reality.

In this paper, we exemplify this approach by applying a foundational ontology named UFO in the design of an agent-oriented modeling language for the ARKnowD (Agent-oriented Recipe for Knowledge Management System Development, read "Arnold") methodology [1]. Given the current stage of research on the agent-oriented paradigm, and the vast availability of methodologies and languages for agent-oriented analysis and design, the methodology presented here is built over existing work. It is our belief that not one methodology possesses all the right characteristics to be applied in a particular domain and/or situation. Instead, these characteristics can often be attained by combining different approaches. ARKnowD explores the combination of Tropos [2] and AORML[3]. Each work is applied in a different development activity, according to their natural propensity: Tropos in Requirements Analysis and AORML in System Design.

The main idea behind the application of UFO regards interpreting the concepts of the languages applied in ARKnowD (i.e. the Tropos's notation and AORML) in terms of the concepts of the ontology. Having understood that, we may simply assume that the concepts of Tropos and AORML which map to the same concept of UFO are equivalent. This has both theoretical and practical implications. For instance, it is common to promote some redesign in the languages due to the deeper analysis provided by the ontology. This analysis enables one to understand better how such languages should model specific domains, which often leads to introducing or suppressing concepts from the original language. In practice, this leads to distinctions in the designed models. Hopefully, these distinctions are not too many, so as to justify the use of this particular language. Such changes generally result in engineering models which are clearer to understand and communicate (thus addressing the requirements mentioned in paragraph one).

Besides the ontological approach, we propose some mapping rules between the notations, inspired in the Model Driven Architecture (MDA) metamodel transformation method [4]. This guarantees a smooth transition from Requirements Analysis to System Design, guiding the developer on the use of the methodology, and facilitating automatic model transformation from one activity to the other.

The focus of this particular paper is to illustrate our approach by: (i) describing the ontological interpretations of Tropos and AORML and (ii) presenting the mapping rules which enable guidance to the designer in producing a detailed design model, whose draft is automatically mapped from the system's architectural model (section 4). Before these core sections, section 2 describes the ontological approach applied to design ARKnowD's language and section 3 provides introductory information on Tropos and AORML, also discussing why these two approaches are appropriate to engineer KM systems. Complementarily, section 5 illustrates the transformation of a Tropos diagram into an AORML diagram, following the proposed mapping rules; finally, section 6 presents some final considerations.

2 Using Foundational Ontologies to Analyze, (re)Design and Combine Conceptual Modeling Languages

Ontologies are recognized as important conceptual tools in Computer Science since the end of the 60s, especially in the areas of conceptual modeling and artificial intelligence [5]. In the past years, we observed an explosion of works related to ontologies in several scientific communities. This is motivated by the potential of ontologies to solve semantic interoperability problems (e.g. application and database integration).

An important point to notice is the difference in meaning of the term "ontology" when used, on the one hand, by the conceptual modeling community and, on the other hand, by the artificial intelligence, software engineering and semantic web communities. In conceptual modeling, the term is used in accordance with its original definition in philosophy, i.e. as a formally and philosophically well-founded model of categories that can be used to articulate conceptualizations in specific engineering models and knowledge domains. Conversely, in the other areas mentioned above, the term ontology has been used to describe: (i) a concrete engineering artifact, designed to serve a specific function, without (or with minimum) concern to theoretical foundational aspects; or (ii) domain models (e.g. biology, finance, logistics etc.) expressed in a knowledge representation language (e.g. RDF, OWL, F-Logic or conceptual modeling language (e.g. UML, EER, ORM).

With respect to the analysis and (re)design of conceptual modeling languages (i.e. the focus of this particular paper), we must understand ontology as in conceptual modeling, i.e. as a theoretical body of knowledge or foundation (that is why we call it *foundational ontology*). Using this foundational ontology as a reference model enables the evaluation, comparison, and identification of correspondences between different modeling languages, in other words, UFO is employed here as a well-founded basis for (1) making explicit the ontological commitments of each modeling language; (2) defining (ontological) real-world semantics for their underlying concepts; (3) providing guidelines for the correct use of these concepts; (4) relating concepts defined in different languages via their ontological semantics. The adequacy of this ontology for our purposes lies on the fact that an important part of UFO (named UFO-C) includes the concepts that are relevant to engineer systems in this particular domain, i.e. knowledge management. More about how UFO-C has been developed may be found in [1].

Finally, we would like to highlight the fact that Ontology-Based approach for combining modeling languages such as the one employed here should not be seen in opposition to a Model-Driven one. Typically, in the former, languages are related either by producing a merged metamodel (using a language such as OMG's MOF), or by defining a transformation model (using a language such as OMG's QVT), which relates the constructs of the complementary languages. Now, one should bear in mind that, in any case, language interoperability is first and foremost a *semantic interoperability problem*. Hence, before we can define a set of transformation rules mapping constructs from a metamodel A to a metamodel B, we must establish the relationship between these constructs. But these relations can only be discovered once we know the relationship between their respective interpretations, i.e., between the elements in the underlying domain conceptualizations which are represented by them. In summary, as demonstrated in this article, the Ontology-Based and Model-Driven approaches can be seen as complementary: while the former focuses on semantic aspects of language interoperability, the latter focuses on syntactic ones.

3 ARKnowD Methodology: Combining Tropos and AORML

ARKnowD's life cycle is composed of four activities, namely requirements elicitation, requirements analysis, architectural design and detailed design. These activities may be iteratively executed up to the point that the solution is modeled in enough detail to enable implementation. Tropos is applied in the first three activities while AORML covers the forth one. More about the lifecycle of this methodology and detailed guidelines on how to proceed to apply it can be found it [1].

3.1 Tropos

The Tropos methodology [2] uses visual modeling language and a set of techniques for goal analysis. Basic constructs of the conceptual modeling language are: *actor*, representing a stakeholder in a given domain, a *role* or a set of roles played by an actor in a given organizational setting, and actor's *goal*, *plan* and *resource*. Moreover, a *dependency* link between pairs of actors allows to model the fact that one actor depends on another in order to achieve a goal, execute a plan, or acquire a resource. Goal analysis is conducted from the point of view of each individual actor; i.e. for each actor's goal, we may consider: means to satisfy it (*means-end relationship*); alternative ways to achieve it (*OR decomposition*); possible sub-goals (*AND decomposition*); goals or plans or resources that can contribute positively or negatively to its achievement (*contribution*). This type of information can be graphically depicted in actor and goal diagrams.

3.2 AORML

The Agent-Object-Relationship (AOR) modeling approach [3] is based on an ontological distinction between active and passive entities, i.e. between *agents* and *objects*. In AORML, an entity can be an *agent, an event, an action (*also specialized in *interaction), a claim, a commitment,* or an *object*. Agent and object form, respectively, the active and passive entities, while actions and events are the dynamic entities of the system model. Commitments and claims establish a special type of relationship between agents. These concepts are fundamental components of social interaction processes and can explicitly help to achieve coherent behavior when these processes are semi or fully automated. Besides AOR models human, artificial and institutional agents. Institutional agents are usually composed of a number of human, artificial, or other institutional agents that act on its behalf. Organizations, such as companies, government institutions and universities are modeled as institutional agents, allowing us to model the rights and duties of their internal agents. For further reference, we refer to [9] and to the AOR website: [http://oxygen.informatik.tu-cottbus.de/aor/].

3.3 Using Tropos and AORML to Engineer KM Systems

KM can be defined as a systematic process for acquiring, organizing and communicating knowledge to all members of an organization, enabling them to be more effective and productive in their work [1,6,7,8]. This process is based on practices and technologies that motivate knowledge exchange, so that knowledge can be replicated and amplified to be used in all points-of-action within the organization.

Perhaps, the main attractive characteristic of Tropos is the fact that it is based in *goal modeling*. According to Nonaka and Takeuchi [8], one of the most important KM references today, one of the main drivers of knowledge creation is the organization's intention, defined as "an organization's aspiration to its goals". This turns goal modeling into an important step towards understanding the strategies of the organization regarding knowledge creation and sharing.

If on one hand, Tropos provides a good abstract view of the organization, on the other hand, this methodology's weakness stems from the fact that it does not provide tools to model agent's interaction and behavior with an appropriate amount of detail. We propose to overcome this limitation by also adopting AORML, an UML-based language to model agent-oriented systems. Understanding how well people interact is crucial to grasp how knowledge flows within the organization. This understanding is also important to enable system agents to go through detailed design, thus being prepared for implementation. AORML offers a set of three types of *interaction diagrams*, modeling agent's interaction protocols as well as their internal behavior. Another strength from AORML is providing deontic modeling constructs such as commitments and claims, which form the basis for the establishment of norms and contracts. Such *normative dimension* is an important one when dealing with agent-mediated KM [7], so as to regulate coordination and operational mechanisms within the organization, while dealing with knowledge creation and dissemination.

4 Ontology-Based Analysis and Design of ARKnowD

As mentioned in section 2, in order to guarantee the consistency of the resulting modeling language, we seek theoretical support in a foundational ontology. The UFO foundational ontology covers concepts such as entities (agents and objects), events and actions, but also what we call social concepts such as plan, action, goal, agent, intentionality, commitment etc. UFO has been assembled mainly based on works from Philosophy and Cognitive Sciences [5,9,10]. The positive outcomes of UFO's application has been multiple. First, because this work provided us with a consistent method to evaluate and combine the Tropos's notation and AORML. But also because it confirmed our intuitions (discussed in section 3.3) that these two applied approaches are indeed suitable for the KM domain. In other words, the concepts which we found suitable to model the KM domain (which are included in the foundational ontology) were mainly the ones covered by these two modeling languages applied in combination.

4.1 UFO

In the sequel, we discuss a fragment of UFO in line with the purposes of this article. For a full discussion regarding this foundational ontology, one should refer to [5,9,10].

We start with the fundamental distinction between universals and individuals. The notion of universal underlies the most basic and widespread constructs in conceptual modeling. Universals are predicative terms that can possibly be applied to a multitude of individuals, capturing the general aspects of such individuals. Individuals are entities that exist instantiating a number of universals and possessing a unique identity.

Further, UFO makes a distinction between the concepts of Endurants and Events (also known as Perdurants). Endurants are individuals said to be wholly present whenever they are present, i.e., they are in time, in the sense that if we say that in circumstance c1 an endurant e has a property P1 and in circumstance c2 the property P2 (possibly incompatible with P1), it is the very same endurant e that we refer to in each of these situations. Examples of endurants are a house, a person, the moon, a hole, an amount of sand. For instance, we can say that an individual John weights 80kg at c1 but 68kg at c2. Nonetheless, we are in these two cases referring to the same individual John. Events (Perdurants), in contrast, are individuals composed by temporal parts, they happen in time in the sense that they extend in time accumulating temporal parts. An example of an Event is a business process. Whenever an Event occurs, it is not the case that all of its temporal parts also occur. For instance, if we consider a business process "Buy a product" at different time instants when it occurs, at each of these time instants only some of its temporal parts are occurring.

A Substantial is an Endurant that does not depend existentially on other Endurants, roughly corresponding to what is referred by the common sense term "Object". In contrast with Substantials, we have Moments (also known as particularized properties and objectified properties). Moments are existentially dependent entities, i.e., for a Moment x to exist, another individual must exist, named is bearer. Examples of Substantials include a person, a house, a planet, and the Rolling Stones; examples of Moments include the electric charge in a conductor, a marriage, a covalent bond as well as mental states such as individual Beliefs, Desires and Intentions (or internal commitments). The last three examples fall in the subcategory of Mental Moments.

UFO also adds distinctions concerning the intentionality of events to this basic core. Examples include the concepts of Action, Action Universal, Action Contribution and Agent.

Actions are intentional events, i.e., events which instantiate a Plan (Action Universal) with the specific purpose of satisfying (the propositional content of) some Commitment of an Agent. The propositional content of a commitment is termed a Goal. Only agents (entities capable of bearing intentional moments) can perform Actions. As events, actions can be atomic (Atomic Action) or complex (Complex Action). While an Atomic Action is an action event that is not composed by other action events, a Complex Action is a composition of at least two basic actions or Participations (that can themselves be atomic or complex).

Participations can themselves be intentional (i.e., Actions) or non-intentional Events. For example, the stabbing of Caesar by Brutus includes the intentional participation of Brutus and the non-intentional participation of the knife. In other words, we take that it is not the case that any participation of an agent is considered an action, but only those intentional participations called Action Contributions.

The category of agents further specializes in Physical Agents (e.g., a person) and Social Agents (e.g., an organization, a society). In an analogous manner, Non-Agentive Substantials (or Objects) can also be categorized as Physical Objects (e.g., cars, rocks and threes) or Social Objects (e.g., a currency, a language, the Brazilian constitution). Agents can also be further specialized into Human Agent, Artificial Agent and Institutional Agent, which can be represented, respectively, by human beings, computationally-based agents and organization or organizational unit (departments, areas and divisions). Institutional Agents are composed by a number of other agents, which can themselves be Human Agents, Artificial Agents or other Institutional Agents.

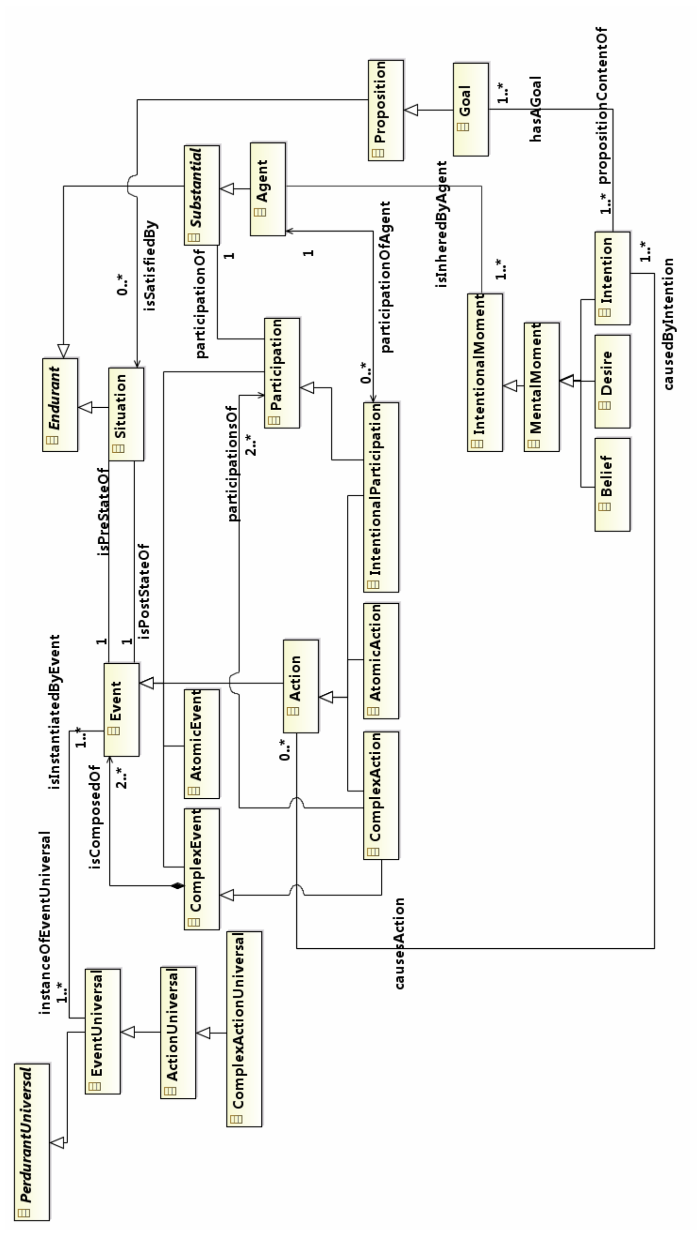


Fig. 1. Fragment of UFO

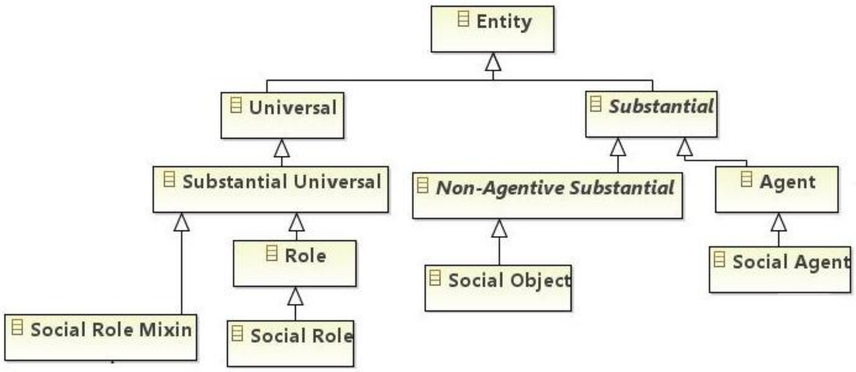


Fig. 2. Fragment of UFO with social aspect

4.2 Applying UFO to Analyze Tropos

We interpret the metaclasses *Actor* and *Role* in Tropos as the concepts of Agent and Social Role in UFO (respectively). An an agent role is defined by the set of social moment universals (commitments and claims implied by the role) [10].

We view Tropos goals as Goals in UFO. Goals in UFO are sets of intended states of affairs of an agent. The relation between an Actor in Tropos and a Goal (through the meta-association *wants*) is interpreted indirectly by making use of the concept of Intention (or Internal Commitment) in UFO, which is a Mental Moment of an Agent. As previously discussed, UFO contemplates a relation between Situations and Goals such that a Situation (or possibly a number of Situations) may satisfy a Goal. In other words, since a Goal is a proposition (the propositional content of an Intention), we have that a particular state of affairs can be the truthmaker of that proposition. This interpretation choice seems to model directly the intention behind the concept of *hardgoal* in Tropos. For the case of *softgoals*, a different analysis must be conducted.

The concept of softgoals does not have a uniform treatment in the Tropos community. Sometimes, softgoals are taken to represent non-functional requirements [11]. In other times, a softgoal is considered as a fuzzy proposition, i.e., one which can be partially satisfied (or satisfied to a certain degree, or yet, *satisficed*) by Situations [12]. We here take a different stance, namely, that a softgoal is one "subjective to interpretation" and "context-specific".

As a consequence of this conception, for the case of softgoals, it seems to be impossible to eliminate a judging agent (collective or individual) from the loop. Thus, instead of considering in the ontology a new *satisfices* relation between Situation and Goal which perhaps should contemplate a fuzzy threshold of satisfaction, we take a different approach. We consider the relation of satisfaction as a ternary relation that can hold between an agent, a goal and situation. An instance of this relation is derived from *the belief of an agent that a particular situation satisfies the goal at hand*. Now, in this view, different agents can have different beliefs about which sets of situations satisfy a given goal. In fact, it is exactly this criterion which seems to capture the aforementioned notion of softgoals and its differentiae w.r.t. hardgoals: (i) a goal G is said to be a hardgoal iff the set of situations that satisfy that goal is necessarily shared by all rational agents; (ii) a goal G is said to be a softgoal iff it is possible that two rational agents X and Y differ in their beliefs to which situations satisfy that goal.

Seeing the distinction between these subcategories of goals under this light, allows us to talk about different levels of "softness" between different formulations of a goal. In one end of the spectrum, each individual agent would have a different belief about which situations satisfy a goal. In the opposite end, we have a hardgoal. In between, we can have communities of agents (or collective agents) of different sizes which share a common belief regarding this set of situations.

The mapping of the Plan concept from Tropos to some UFO concept is established in a direct manner. In section 3.1, we stated that a Plan in Tropos is a specific way of doing something to satisfy some Goal (or *satisfacing* some Softgoal). From the UFO ontology (section 4.1), we have that an Action (instance of an Action Universal) is an intentional event performed by agents with the purpose of achieving goals. Consequently, the Tropos *Plan* construct can be interpreted as an Action Universal.

In Tropos, goals can be further structured by using different types of relations, namely, AND-decomposition and OR-decomposition. Since Goals are taken here to be propositions, if we have that goals $G_1 \ldots G_n$ AND-decompose goal G_0, this relation should be interpreted as: $(G_0 \leftrightarrow (G_1 \wedge G_2 \wedge \ldots \wedge G_n))$. In an analogous manner, and OR-decomposition $G_1 \ldots G_n$ of goal G_0 should be interpreted as: $(G_0 \leftrightarrow (G_1 \vee G_2 \vee \ldots \vee G_n))$. Here once more, these relations reflect logical relations between propositions and, accordingly, are independent of an Agent's point of view (contra Fig.2).

We have offered an ontological analysis of the relation of Dependency in Tropos elsewhere [10]. In that paper, we show that Tropos overloads in the same construct the two different (ontological) relations of Dependency and Delegation, which constitutes another case of construct overload in the language. As discussed in depth there, these relations belong to different ontological categories: whilst the former is an example of a formal relation, the latter is one a material relation. To put it baldly, agent X *depends on* agent Y for goal G iff G is a goal of X, X cannot achieve G, and Y can achieve G. Notice that in this case, agents X and Y do not even have to be aware of this dependency. In contrast, if agent X delegates goal G to agent Y then: there is a social commitment c from Y to X; G is the propositional content of c.

The remaining relationship types from Tropos (namely, means-end and contribution) remain to be analyzed in detail. This remains as future work and should thus bring new theoretical and practical implications.

4.3 Applying UFO to Analyze AORML

The primitives contained in AORML and the ontological categories in UFO bear a rather straightforward relation to each other. Here, due to space limitations we refrain from presenting an AORML metamodel and focus on the fragment of this language which is germane to the purposes of this article. The notions of agent, action, event, commitment and claim in AORML are directly mapped to their counterparts in UFO. The notion of Object in AORML is mapped to the one of Non-Agentive Substantial (or Object) in UFO. An interaction in AORML is interpreted as an Action in UFO, i.e., interactions in AORML can also represent a single-contribution of an Agent in joint action. Finally, a relationship in AORML is interpreted as a relation universal in UFO (further specialized in both formal relation and material relation) [1].

4.4 Combining Tropos and AORML through Metamodel Transformation

Having clarified the semantics of the modeling constructs through interpretation in terms of UFO, we can establish the correspondence between the constructs in each of the identified fragments of Tropos and AORML (see in Table 1).

Table 1. Mapping Tropos into AORML

Tropos Concepts	AORML Constructs
actor	agent
plan	interaction
resource	object
dependency	relationship
delegation	relationship and commitment
resource acquisition	relationship and commitment

In this work, we apply the MDA's metamodel transformation technique, which requires a mapping from the modeling constructs of the source (the Tropos' notation) to the destiny language (AORML). In other words, mapping concepts as prescribed by Table 1 has practical implications in designing the system's model. For example, a Tropos's *plan* may be mapped into AORML's *interaction* concept. In practice, for each plan in a Tropos model, there can be one or more **AOR Interaction Sequence Diagram**, modeling the interactions of the agents participating in this plan (i.e. agents having the plan, or being connected to it by a dependency link). Another interesting illustration comes from the differentiation we introduced between Tropos' *dependency* and *delegation*. The former only maps into an AORML relationship while the latter maps both to a relationship and a commitment. Conceptually, this should be clear from section 4. However, in practice, this leads to the following distinctions: both for Tropos dependency and delegation, an association link may be depicted between these agents in an **AOR Agent Diagram**, typically used for information modeling. Now, besides this association delegations also lead to the establishment of an AORML c*ommitment/claim pair* between the (delegate and delegator) agents. This construct is usually depicted in interaction modeling, using one or more types of **AOR interaction diagrams**.

Note that one of the most important entities in Tropos, i.e. the concept of *goal* is not mapped into AORML. This is not a contradiction. Conversely, it relates to the fact that ARKnowD applies goal modeling exclusively for requirements analysis and architectural design. On detailed design, all goals have already been dealt with. For instance, goals may have been fulfilled or abandoned. But most commonly, goal analysis leads to the delegation of unsolved goals to new or old agents, who are either part of the organization or a new information system. And finally, concrete plans are assigned to goals with the purpose of accomplishing them. Consequently, when the detailed design activity starts, plans should be modeled rather than goals. As observed in Table 1, plan modeling may be done through the use of AOR Interaction Sequence

Diagrams, which detail the protocol of communication between agents to realize a specific sequence of actions. In the end, we do not, however, loose the connection to the goals initially modeled in during requirements analysis. This is still traceable through the plan trees linked to each of these goals.

After conceiving the mapping rules of Table 1, it is possible to automate the metamodel transformation between the two languages, by implementing these rules. Aiming at providing automated support to ARKnowD, we started to integrate AORML into an existing Tropos modelling tool named TAOM4E (http://sra.itc.it/tools/taom4e/), implementing the mapping of a Tropos Actor Diagram into an AORML agent Diagram. For that, we used a transformation engine named Tefkat (http://tefkat.sourceforge.net/), Basically, Tefkat receives as input the metamodels of the two modeling languages (i.e. the metamodels of the Tropos language and AORML), along with the source Tropos model developed with the use of TAOM4E. The mapping between the two metamodels is directly implemented using Tefkat's declarative language. The result is an AORML model. Future work remains on the remaining mappings, so as to deliver a modelling tool which enables full design using ARKnowD.

5 Working Example

In this section, we present a simple example of the use of ARKnowD, with the main purpose of illustrating the transformation between the notations of Tropos and AORML. We find the conference review process an appropriate scenario to exemplify ARKnowD, because this is a well-known setting for the academic community. For space limitation, we just exemplify the use of two diagrams (one of each language) that were targeted in the aforementioned automation initiative. For a full description of this working example, including other diagrams, as well as for a more complex scenario of application of ARKnowD in the Knowledge Management domain, please refer to [1].

Figure 3 presents a Tropos actor diagram, depicting the main agents of the scenario, along with some goal and resource dependencies between them. The diagram shows that the scenario involves the participation of four agents, namely the Conference Chair, the PC Chair, the Paper Author and the PC Member. For realizing the conference, the Conference Chair depends on the Paper Author to submit papers that will be selected for presentation in the conference (submitting paper goal). For this papers selection, the Conference Chair delegates to the PC Chair the responsibility of selecting the best papers to be published in the conference proceedings (selecting proceedings' papers goal). The PC Chair and the Paper Author have a mutual relationship. While the PC Chair wants to acquire papers submitted by the Paper Author (submitted paper resource), the Paper Author delegates to the PC Chair the goal of having his paper reviewed as part of the papers selection process (having paper reviewed goal). However, the PC Chair does not review all papers on his own. For that, he relies on PC Members (reviewing papers goal). For accomplishing this goal, the PC Member must receive the papers assigned to them (assigned paper resource), along with the review form (review form resource) from the PC Chair.

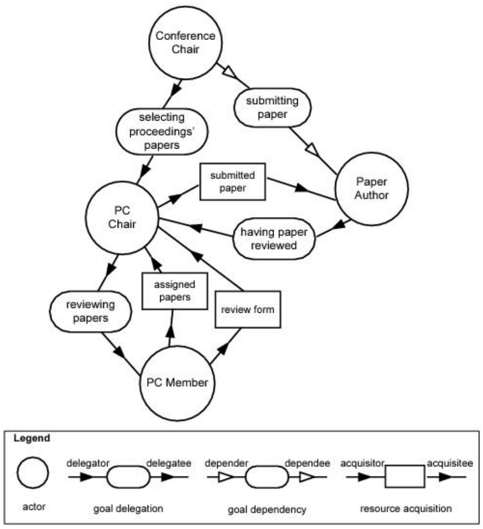


Fig. 3. Tropos Actor Diagram

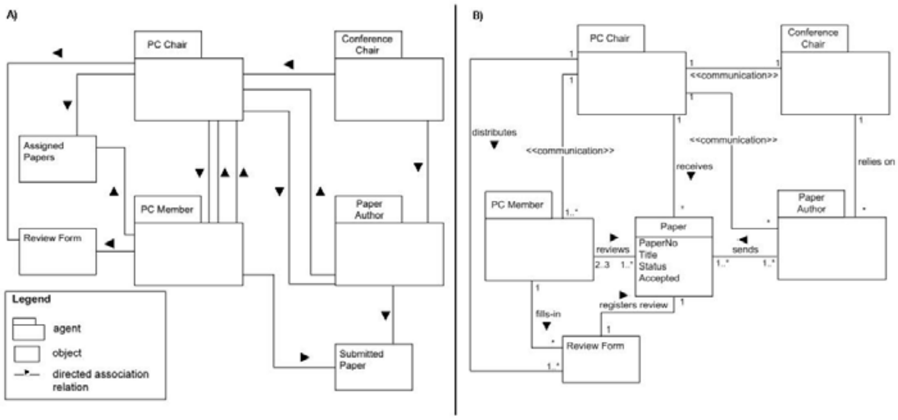


Fig. 4. AOR agent diagrams: (A) automatically generated from previous Tropos actor diagram and (B) finalized, after designer's edition

At this point, we can already exemplify the first transformation. Figure 4(A) depicts an AOR Agent Diagram (AD) that can be automatically generated with basis on the goal diagram of Fig. 3, using the transformation rules described in Table 1.

This figure depicts the agents and objects of the scenario, respectively transformed from the Tropos agent and resource constructs. Besides the scenario's entities, the diagram also depicts the relations between them, converted from the dependencies, delegations and acquisitions shown in the previously presented Tropos actor diagram. Both the number and direction of the relations between agents are inferred from the number and directions of the dependency, delegation and acquisition links on the actor diagram. For instance, between PC Chair and PC Member, there are three relations, corresponding to the two acquisitions and one delegation previously depicted in Fig. 3, and following the same directions of such links. Although this first automatic AD is truthful to our scenario, some modifications may be necessary for enabling its best use in practice. This diagram can then be revised and modified, resulting in the AD of Fig. 4(B). In this second AD, two objects from the AD of Fig.4(A), namely Submitted Paper and Assigned Paper were merged into the Paper object. This comes from the realization that the previously depicted resources on the Tropos actor diagram actually referred to the same object, in two different states (i.e. 'submitted' and 'assigned'). Hence, the two objects originated a single one, and such state is now given by the status attribute in the Paper object. In addition to that change, multiple relations between agents were reduced to one (as a result of a choice made by the designer. In other situations, multiple relations may be considered desirable, thus being maintained) and all relations were named. Finally, some associations between two agents were substituted by a specific type of relation, named communication relation (note the communication stereotype, an extension introduced by AORML). Besides being related by associations, agents typically relate through communication relations, which indicate that they interact to accomplish their goals. Typically, communication relations occur among agents that previously delegated goals or tasks, or acquired resources from one another. In other words, for a delegation or an acquisition to occur, agent A must explicitly interact with agent B, either to ask him/her to accomplish some goal or execute a task on his/her behalf, or to acquire a resource controlled by agent B.

Note also that the diagram of Fig. 4(B) presents the cardinalities (not present in the type of model of Fig. 4(A)) of all agents and objects of the scenario. In the case depicted here, only association relations are necessary among the scenario's entities. In other cases, generalization and composition relations may be necessary. In general, all UML relations may be normally used in the AOR AD.

6 Final Considerations

This paper described our approach to design an engineering language to the ARKnowD methodology, which combines two distinct agent-oriented software engineering approaches, namely Tropos and AORML. For mapping the two notations, a theoretical analysis was made with the use of the UFO foundational ontology. Moreover, an MDA-inspired transformation method was used and partially implemented in an agent-oriented modeling tool named TAOM4E, currently under development. Ongoing work in that respect includes analysing further concepts and relationships from the source languages. Moreover, we are currently developing case studies in a real setting. The results of this case study should point out to other directions regarding the evolution of the ARKnowD methodology.

Furthermore, our research group has been involved in several initiatives applying ontological foundations to enable modelling languages evaluation and (re) design. The most related ones regard: a) the combination of goal modelling (Tropos) and business process modelling (ARIS-EPC) to enable a comprehensive strategic analysis before stepping into business process engineering within organizations [13]; b) analyzing and exposing the semantics behind Software Process Reference Models [9].

Acknowledgments. This research has been partially supported by FAPES (Grants #37274554/07 and #45444080/09) and CNPq (Grant #481906/2009-6). The second author of this chapter holds the CNPq (Brazilian National Research Council) Productivity Grant #309382/2008 4. We are grateful to João Paulo A. Almeida, Evellin C. Cardoso and Paulo Sérgio Santos Jr. for their valuable contributions to this work.

References

1. Guizzardi, R.S.S.: Agent-oriented Constructivist Knowledge Management. PhD Thesis, University of Twente, The Netherlands (2006)
2. Bresciani, P., Giorgini, P., Giunchiglia, F., Mylopoulos, J., Perini, A.: Tropos: An Agent-Oriented Software Development Methodology. Int. J. of Autonomous Agents and Multi Agent Systems 8(3), 203–236 (2004)
3. Wagner, G.: The Agent-Object-Relationship Meta-Model: Towards a Unified View of State and Behavior. Information Systems 28(5), 475–504 (2003)
4. Miller, J. and Mukerji, J.: MDA Guide Version 1.0.1, omg/2003-06-01 (2003), `http://www.omg.org/docs/omg/03-06-01.pdf`
5. Guizzardi, G.: Ontological Foundations for Structural Conceptual Models. PhD Thesis, University of Twente, The Netherlands (2005)
6. Dignum, V.: A Model for Organizational Interaction: Based on Agents, Founded in Logic. PhD thesis, Utrecht University, The Netherlands (2004)
7. Dignum, V.: An Overview of Agents in Knowledge Management. Technical Report UU-CS-2004-017, Inst. Information and Computing Sciences, Utrecht University, The Netherlands (2004)
8. Nonaka, I., Takeuchi, H.: The Knowledge Creating Company: How Japanese Companies Create the Dynamics of Innovation. Oxford University Press, New York (1995)
9. Guizzardi, G., Guizzardi, R.S.S., Falbo, R.A.: Grounding Software Domain Ontologies in the Unified Foundational Ontology (UFO): The case of the ODE Software Process Ontology. In: Proc. of the Iberoamerican Conference on Software Engineering (CIbSE 2008), Recife, Brazil (2008)
10. Guizzardi, R.S.S., Guizzardi, G.: From Tropos to AORML, Using a Foundational Ontology. In: Giorgini, P., Maiden, N., Mylopoulos, J., Yu, E. (eds.) Tropos/i*: Applications, variations and Extensions, Cooperative Information Systems Series. MIT Press, Cambridge (forthcoming)
11. Chung, L., Nixon, B., Yu, E., Mylopoulos, J.: Non-functional requirements in Software Engineering. Kluwer Academic Publishers, Dordrecht (2000)
12. Letier, E., van Laamsweerde, A.: Reasoning about Partial Goal Satisfaction for Requirements and Design Engineering. ACM SIGSOFT Software Engineering Notes 29(6) (2004)
13. Cardoso, E.C.S., Almeida, J.P.A., Guizzardi, G., Guizzardi, R.S.S.: Eliciting Goals for Business Process Models with Non-Functional Requirements Catalogues. In: 10th International Workshop on Business Process Modeling, Development and Support, CAISE 2009, Amsterdam, vol. 29, pp. 33–45 (2009)

A Model of Independence and Overlap for Transactions on Database Schemata

Stephen J. Hegner

Umeå University, Department of Computing Science
SE-901 87 Umeå, Sweden
hegner@cs.umu.se
http://www.cs.umu.se/~hegner

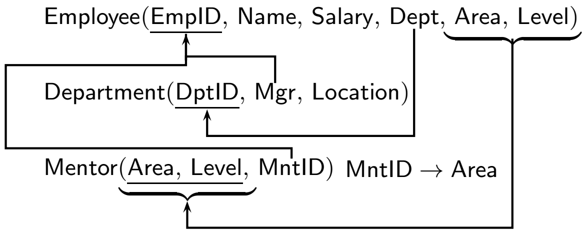
Abstract. Traditional models of support for concurrent transactions invariably rely upon a notion of serializability, which involves not only complex scheduling, but also primitives (such as locks) for requiring transactions to wait, as well for aborting a transaction and forcing it to re-run. For batch transactions, this approach is often the most reasonable. On the other hand, for interactive transactions, only a very limited amount of waiting and aborting is tolerable, and so minimizing their occurrence, even at the cost of increased analysis of the transactions themselves, is warranted. In this work, a systematic study of independence for transactions, without any explicit serialization, is initiated. Each transaction operates on a view of the main schema, and each such view is partitioned into a write region and a read-only region. For a set of transactions to run concurrently, their views may overlap only on their read-only regions. These regions need not be specified explicitly; rather, they are defined naturally using a component-based model of the main schema. Furthermore, when two transactions do conflict, because their views overlap on write regions, the precise point of conflict is immediately identified. To illustrate the utility of the framework, the case of relational schemata governed by the most common types of constraints in practice — functional and foreign-key dependencies — is developed in detail.

1 Introduction

Support for concurrent transactions has long been a crucial feature of database-management systems. Central to all approaches is a notion of conflict. Typically, the database is modelled as a set $X = \{x_1, x_2, \ldots, x_n\}$ of data objects, each of which may be read and modified by transactions. Two transactions are in potential conflict if they operate on the same data object in certain ways. In the case of a potential conflict, the standard criterion for admissibility is that the transactions involved be interleaved in such a way that the the result is equivalent to that obtained were they not interleaved; so-called *serializability* [3, Ch. 22] [15, Ch. 2]. For enforcement, the scheduler may require a transaction to wait (e.g., via locking) for a resource which is held by another transaction, and in the worst case it may require a transaction to abort and re-run.

In recent years, the need to support interactive transactions has grown enormously. With humans in the loop, techniques which impose long waits (such as locking) are clearly undesirable, if not outright unacceptable. Therefore, such actions must be used sparingly, if at all. Of course, there is no magical way to avoid conflict of concurrent operations. On the other hand, with interactive transactions, additional overhead of a few milliseconds or even a few seconds is a reasonable tradeoff for reduced delays and aborts, so it is realistic to work with a more complex model of data objects, providing a finer analysis of how the operations of distinct transactions interact and resulting in fewer conflicts and a better model of identifying those conflicts which do occur. The goal of this work is to provide such a model. The basic idea is that data objects are not just sets of tuples, but rather are defined by objects which are structured views. Each such view-object has a number of sub-views which define a read-only region. Two objects can never entail a conflict of transactions if those objects overlap at most on their read-only regions. There is furthermore a calculus of combination of these objects which creates larger objects. If all objects which contain a given read-only region are combined, then that region becomes writable. The formal model is based database schema components, as developed in [8]. As such, it distinguishes itself from other work on similar topics, such as [18] and [17], which focus more on transaction primitives. On the other hand, it also distinguishes itself from semantic approaches, such as [12], in which increased concurrency is obtained by modelling the the data objects as abstract data types with explicitly defined operations. In the work reported here, the data objects have no structure beyond the definitions of write and read-only regions.

In describing the work in this paper, the term *claim*, rather than *lock*, will be used to describe a data object which has been assigned to a transaction. The reason is that this paper is not about locking or serialization, it is rather about modelling independence and conflict. How the latter is prevented or resolved is not the focus. Locking is one way to prevent conflict, but solutions which involve negotiation, for example, may also be appropriate in the context of interactive transactions. Before proceeding to the development of the main ideas, it is useful to illustrate the basic idea via a running example, which will also be used in other parts of the paper. The relational schema $\mathbf{E}_0$ has the three relations and integrity constraints identified in Fig. 1. The keys are underlined,



Employee(EmpID, Name, Salary, Dept, Area, Level)
Department(DptID, Mgr, Location)
Mentor(Area, Level, MntID)
MntID → Area

Fig. 1. Diagrammatic representation of the running example $\mathbf{E}_0$

and foreign keys are indicated by the arrows from the foreign key to the associated primary key. Thus, $\mathsf{Department}[\mathsf{Mgr}] \subseteq \mathsf{Employee}[\mathsf{EmpID}]$, $\mathsf{Mentor}[\mathsf{MntID}] \subseteq \mathsf{Employee}[\mathsf{EmpID}]$, and $\mathsf{Employee}[\mathsf{Area}, \mathsf{Level}] \subseteq \mathsf{Mentor}[\mathsf{Area}, \mathsf{Level}]$. The functional dependency $\mathsf{MntID} \rightarrow \mathsf{Area}$ is shown explicitly because it is not a key constraint. All relations are in 3NF (third normal form). The relation Mentor, constrained by both of the FDs (functional dependencies) $\{\mathsf{Area}, \mathsf{Level}\} \rightarrow \mathsf{MntID}$ and $\mathsf{MntID} \rightarrow \mathsf{Area}$, is a classical example of a relation which cannot be decomposed further into BCNF (so that the only FDs are key dependencies) while retaining a cover of the governing dependencies [4, Sec. 10.5].

In most approaches to defining data objects for transactions, even those which forward using so-called predicate locks such as [5] and [14], the data objects always return a set of tuples from a relation. However, it is advantageous, and also a natural feature of the theory developed here, to allow claim sets to be defined by views, since two transactions may be able to update different fields of the same tuple. Consider two simple transactions. T_1 is to give a 5% raise to each employee in the Research department, and T_2 is to change the names of employees in the set NSet (because they married, say). It is clear that these two operations may be carried out concurrently, even though they may operate on the same tuples (in the case that NSet contains employees who work in the Research department). Formally, T_1 operates on the view defined by $\sigma_{\mathsf{Dept}=\mathrm{Research}}\langle\mathsf{Employee}[\mathsf{EmpID}, \mathsf{Salary}, \mathsf{Dept}]\rangle$; i.e., the selection to the Research department of the projection $\mathsf{Employee}[\mathsf{EmpID}, \mathsf{Salary}]$, while T_2 operates on the view defined by $\sigma_{\mathsf{Name}=\mathrm{NSet}}\langle\mathsf{Employee}[\mathsf{EmpID}, \mathsf{Name}]\rangle$. These two data objects overlap on their read claims, but not on their write claims. Specifically, the write claim of T_1 involves only the values of the Salary field, while that for T_2 involves only the name field. Their read claims overlap on the EmpIDs of their common tuples. Since neither transaction may alter these EmpIDs, they may be shared. More complex operation on this example schema and others will be considered in that which follows.

The remainder of the paper is composed of two main sections. In Section 2, the formal model is developed, independently of any specific data model. In Section 3, these ideas are applied to the relational model, constrained by FDs and foreign-key dependencies (FKDs). The basic components identified via vertical decomposition (i.e., projections) are augmented with a further decomposition into horizontal components, defined by selection. Finally, Section 4 provides a summary and indication of possible further directions.

2 Component-Based Independent Updates

In this section, the fundamental ideas of the model for complex update objects for transactions are developed. As they do not depend upon any particular data model, they are developed within a general framework of set-based schemata. The basic definitions of schema, morphism, view, complement, and the like parallel those developed in [6], to which the reader is referred for details. A summary of some of these ideas may be found in [7, Sum. 2.1] as well. The basic ideas for the component-based concepts are based upon those in [8]. The relational model will nevertheless be used for some examples. Because it is so widely known, the standard notation and terminology surrounding it, as may be found in textbooks such as [4] and [13], will not be reviewed but rather assumed.

Definition 2.1 (Database schemata, views, and updates). A *set-based database schema* $\mathbf{D}$ is one for which a finite set $\mathsf{LDB}(\mathbf{D})$ of *legal database states* is given. In the relational model, $\mathsf{LDB}(\mathbf{D})$ is the set of states which satisfy the integrity constraints of the schema. A *morphism* $f : \mathbf{D}_1 \to \mathbf{D}_2$ of set-based schemata is given by a function $\mathsf{LDB}(f) : \mathsf{LDB}(\mathbf{D}_1) \to \mathsf{LDB}(\mathbf{D}_2)$. Since no confusion can result, the qualifier LDB will usually be dropped; i.e., $f : \mathsf{LDB}(\mathbf{D}_1) \to \mathsf{LDB}(\mathbf{D}_2)$. For the rest of this section, unless stated specifically to the contrary, the term *database schema* will mean *set-based database schema*.

A *view* of the database schema $\mathbf{D}$ is a pair $\Gamma = (\mathbf{V}, \gamma)$ in which $\mathbf{V}$ is a database schema and $\gamma : \mathbf{D} \to \mathbf{V}$ is a database morphism for which the underlying function $\gamma : \mathsf{LDB}(\mathbf{D}) \to \mathsf{LDB}(\mathbf{V})$ is surjective. The surjectivity ensures that every state of the view schema $\mathbf{V}$ is the image of some state of the main schema $\mathbf{D}$. The equivalence relation $\mathsf{Congr}(\Gamma) = \{(M_1, M_2) \in \mathsf{LDB}(\mathbf{D}) \times \mathsf{LDB}(\mathbf{D}) \mid \gamma(M_1) = \gamma(M_2)\}$ is called the *congruence* of Γ. If $\Gamma_1 = (\mathbf{V}_1, \gamma_1)$ and $\Gamma_2 = (\mathbf{V}_2, \gamma_2)$ are views with $\mathsf{Congr}(\Gamma_1) \subseteq \mathsf{Congr}(\Gamma_2)$, then Γ_2 may be thought of as a smaller view than Γ_1, in that Γ_1 preserves more information about the state of $\mathbf{D}$ than does Γ_2. The special notation $\Gamma_2 \sqsubseteq_{\mathbf{D}} \Gamma_1$ will be used to indicate that $\mathsf{Congr}(\Gamma_1) \subseteq \mathsf{Congr}(\Gamma_2)$. To make this idea more precise, given that $\Gamma_2 \sqsubseteq_{\mathbf{D}} \Gamma_1$, define the function $\lambda\langle \Gamma_1, \Gamma_2 \rangle : \mathbf{V}_1 \to \mathbf{V}_2$ by $N \mapsto \gamma_1(M)$ for any $M \in \gamma_1^{-1}(N)$. This function is well defined, since if $M_1, M_2 \in \gamma_1^{-1}(N)$, then $\gamma_2(M_1) = \gamma_2(M_2)$, owing to the fact that $\mathsf{Congr}(\Gamma_1) \subseteq \mathsf{Congr}(\Gamma_2)$. Furthermore, Γ_2 may be regarded as a view $\Lambda(\Gamma_1, \Gamma_2) = (\mathbf{V}_2, \lambda\langle \Gamma_1, \Gamma_2 \rangle)$ of $\mathbf{V}_1$. As a concrete example, let $\mathbf{E}_1$ be the relational schema with the single relation schema $R[ABC]$, constrained by the functional dependencies (FDs) $A \to C$ and $B \to C$. Let $\Pi_{AB}^{\mathbf{E}_1} = (\mathbf{E}_{1_{AB}}, \pi_{AB}^{\mathbf{E}_1})$ be the view which defines the projection of $R[ABC]$ onto AB; the single relation symbol of $\mathbf{E}_{1_{AB}}$ is $R[AB]$. Similarly, let $\Pi_{B}^{\mathbf{E}_1} = (\mathbf{E}_{1_B}, \pi_{B}^{\mathbf{E}_1})$ be the view which defines the projection of $R[ABC]$ onto B; the single relation symbol of $\mathbf{E}_{1_B}$ is $R[B]$. The function $\lambda\langle \Pi_{AB}^{\mathbf{E}_1}, \Pi_{B}^{\mathbf{E}_1} \rangle$ is just the projection $\pi_B^{\mathbf{E_{AB}}_1}$ of $R[AB]$ onto $R[B]$, with the relative view $\Pi_B^{\mathbf{E}_{1_{AB}}} = (\mathbf{E}_{1_B}, \pi_B^{\mathbf{E_{AB}}_1})$.

It is also important to note that each equivalence relation $r \subseteq \mathsf{LDB}(\mathbf{D}) \times \mathsf{LDB}(\mathbf{D})$ defines a set-based view $\Gamma_{[r]} = (\mathbf{V}_{[r]}, \gamma_{[r]})$ with $\mathsf{LDB}(\mathbf{V}_{[r]}) = \mathsf{LDB}(\mathbf{D})/r$, the blocks of the equivalence relation r, and with $\gamma_{[r]} : M \mapsto \{M' \mid r(M, M')\}$. Indeed, the congruence of a set-based view $\Gamma = (\mathbf{V}, \gamma)$ characterizes it up to a renaming of the elements of $\mathsf{LDB}(\mathbf{V})$. More formally, a *morphism* $h : \Gamma_1 \to \Gamma_2$ of views is given by a morphism $h : \mathbf{V}_1 \to \mathbf{V}_2$ on the underlying schemata with the property that $h \circ \gamma_1 = \gamma_2$. In the case of set-based views, h is bijective (hence an isomorphism) iff $\mathsf{Congr}(\Gamma_1) = \mathsf{Congr}(\Gamma_2)$.

The *zero view* on $\mathbf{D}$, denoted $\mathsf{ZView}_{\mathbf{D}}$, is a view whose congruence is $\mathsf{LDB}(\mathbf{D}) \times \mathsf{LDB}(\mathbf{D})$. Thus, its schema has only one state, and so it conveys no information about the state of the main schema. Dually, the *identity view* $\mathsf{IdView}_{\mathbf{D}} = (\mathbf{D}, \mathsf{IdMor}_{\mathbf{D}})$ on $\mathbf{D}$ is the view which is the identity on $\mathsf{LDB}(\mathbf{D})$. Each will be useful in certain constructions.

The *join* of two set-based views $\varGamma_1$ and $\varGamma_2$ is the set-based view (unique up to isomorphism) whose congruence is $\mathsf{Congr}(\varGamma_1) \cap \mathsf{Congr}(\varGamma_2)$. It is denoted $\varGamma_1 \sqcup \varGamma_2$. In the case of relational views, the join may be constructed explicitly by taking the view schema to be the (disjoint) union of the schemata of the two views. See [9, Def. 4.3] for details. Since the join is associative, the definition extends naturally to an arbitrary finite set. The join of a finite set $S = \{\varGamma_1, \ldots, \varGamma_n\}$ of views is denoted $\varGamma_1 \sqcup \ldots \sqcup \varGamma_n$, or $\bigsqcup_{i=1}^{n} \varGamma_i$, or just $\bigsqcup S$.

An *update* on the schema $\mathbf{D}$ is a pair $(M_1, M_2) \in \mathsf{LDB}(\mathbf{D}) \times \mathsf{LDB}(\mathbf{D})$, with M_1 the old state and M_2 the new state. The set of all updates on $\mathbf{D}$ is denoted $\mathsf{Updates}(\mathbf{D})$. Let $u = (N_1, N_2) \in \mathsf{Updates}(\mathbf{V})$ be an update on the view schema $\mathbf{V}$, and let $M_1 \in \mathsf{LDB}(\mathbf{D})$ with $\gamma(M_1) = N_1$. A *reflection* (or *translation*) of the view update u to $\mathbf{D}$ relative to M_1 is an update $u' = (M_1, M_2)$ on $\mathbf{D}$ with the property that $\gamma(M_2) = N_2$.

The update $(M_1, M_2) \in \mathsf{Updates}(\mathbf{D})$ is *constant on $\varGamma$* if $\gamma(M_1) = \gamma(M_2)$.

Notation 2.2. Throughout this section, unless specifically stated to the contrary, $\mathbf{D}$ will be taken to be a set-based database schema. Furthermore, $\varGamma = (\mathbf{V}, \gamma)$, as well as $\varGamma_x = (\mathbf{V}_x, \gamma_x)$ for any subscript x, will be taken to be a set-based view over $\mathbf{D}$.

If $\mathbf{E}_x$ is a relational schema with single relation $R[\mathbf{U}]$ for some set $\mathbf{U}$ of attributes, and $\mathbf{W} \subseteq \mathbf{U}$, then $\varPi_{\mathbf{W}}^{\mathbf{E}_x} = (\mathbf{E}_{x\mathbf{W}}, \pi_{\mathbf{W}}^{\mathbf{E}_x})$ is the projection view on $\mathbf{W}$ whose relation symbol is denoted by $R[\mathbf{W}]$.

Definition 2.3 (Complementary sets and pairwise definability). Let $\mathfrak{C} = \{\varGamma_1, \varGamma_2, \ldots, \varGamma_n\}$ be a finite set of views of $\mathbf{D}$. Call $\mathfrak{C}$ a *complementary set* if $\bigsqcup_{i=1}^{N} \varGamma_i = \mathsf{IdView}_{\mathbf{D}}$. If $n = 2$, $\mathfrak{C}$ is also called a *complementary pair*. The *decomposition mapping* for $\mathfrak{C}$ is $\gamma_1 \times \ldots \times \gamma_k : \mathsf{LDB}(\mathbf{D}) \to \mathsf{LDB}(\mathbf{V}_1) \times \ldots \times \mathsf{LDB}(\mathbf{V}_k)$, given on elements by $M \mapsto (\gamma_1(M), \ldots, \gamma_k(M))$. It is immediate that $\mathfrak{u}$ is a complementary set iff $\gamma_1 \times \ldots \times \gamma_k$ is injective. Thus, a complementary set of views defines a way to recover the state of the main schema from the combined states of the views. In classical terms, it defines a lossless decomposition.

Think of $\mathfrak{C}$ as representing the set X of data objects in the concurrency model, as identified in Section 1. A transaction T wishes to effect an update u which involves some subset $S_T \subseteq \mathfrak{C}$; formally, u is specified as an update on the schema of $\bigsqcup S_T$. Ideally, other transactions could then be allowed to update the views not in S_T. Unfortunately, for most realistic schemata, it is not possible to find a useful set $\mathfrak{C}$ of views whose members are *independent* in the sense that each may be updated without affecting the states of the others. Rather, the elements of such a set of views typically overlap, and the updates which are permitted must respect that overlap. The necessary additional condition is found in a classical result in the theory of database decomposition. In [2], a number of "desirable" aspects of universal relational schemata are presented. One of these is *pairwise definability.* Let $\mathbf{U}$ be a finite set of attributes, and consider $\mathbf{E}_{\mathbf{U}}$, as defined in Notation 2.2, constrained by some set $\mathcal{F}$ of dependencies. Let $Z = \{\varPi_{\mathbf{U}_i}^{\mathbf{E}_{\mathbf{U}}} \mid 1 \leq i \leq k\}$. be a finite set of projections of $\mathbf{E}_{\mathbf{U}}$ and assume that Z is a complementary set of views. Call a sequence $S = \langle M_1, M_2, \ldots, M_k \rangle$ with

$M_i \in \mathsf{LDB}(\mathbf{E}_{\mathbf{U}_i})$, $1 \leq i \leq k$, *compatible* for Z if there is an $M \in \mathsf{LDB}(\mathbf{E}_{\mathbf{U}})$ with the property that $\pi^{\mathbf{E}_{\mathbf{U}}}_{\mathbf{U}_i}(M) = M_i$ for $1 \leq i \leq n$, and call S *pairwise compatible* for Z if whenever $\mathbf{U}_i \cap \mathbf{U}_j \neq \emptyset$, then $\pi^{\mathbf{E}_{\mathbf{U}_i}}_{\mathbf{U}_i \cap \mathbf{U}_j}(M_i) = \pi^{\mathbf{E}_{\mathbf{U}_j}}_{\mathbf{U}_i \cap \mathbf{U}_j}(M_j)$. Call Z *pairwise definable* if every pairwise compatible set for Z is compatible for Z; that is, agreement on the overlapping columns is sufficient to ensure consistency. Within this model, there is a simple characterization of pairwise definability; namely, that a cover of $\mathcal{F}$ embed into the members of Z. For a complementary pair, a proof may be found in [6, 2.17]; the extension to larger sets is straightforward.

To extend this idea to the general case of a set $\mathfrak{C} = \{\Gamma_1, \Gamma_2, \ldots, \Gamma_n\}$ of views of $\mathbf{D}$, first define a sequence $S = \langle M_1, \ldots, M_n \rangle$ of states with $M_i \in \mathsf{LDB}(\mathbf{V}_i)$ for $1 \leq i \leq k$ to be *compatible for* $\mathfrak{C}$ if there is an $M \in \mathsf{LDB}(\mathbf{D})$ with the property that $\gamma_i(M) = M_i$ for $1 \leq i \leq n$. To obtain a notion of pairwise compatibility, the idea is to specify a set $\mathfrak{P}$ of views on which the elements of $\mathfrak{C} = \{\Gamma_1, \Gamma_2, \ldots, \Gamma_n\}$ must agree, called the set of *ports*. More precisely, say that S is *pairwise compatible for* $\mathfrak{C}$ *with respect to* $\mathfrak{P}$ if $\lambda\langle \Gamma_i, \Gamma \rangle(M_i) = \lambda\langle \Gamma_j, \Gamma \rangle(M_j)$ for every pair $\{\Gamma_i, \Gamma_j\} \subseteq \mathfrak{C}$ and every $\Gamma \in \mathfrak{P}$ for which $\Gamma \sqsubseteq_{\mathbf{D}} \Gamma_i$ and $\Gamma \sqsubseteq_{\mathbf{D}} \Gamma_j$. Say that $\mathfrak{C}$ is *pairwise definable via* $\mathfrak{P}$ if every pairwise compatible set is compatible. In the relational example above, the set of ports is $\{\Pi^{\mathbf{E}_{\mathbf{U}}}_{\mathbf{W}} \mid \mathbf{W} \subseteq \mathbf{U}\}$; that is, the set of all projections. There is a useful visualization of pairwise definability, using the conventions introduced in [8]. Each main view in $\mathfrak{C}$ (corresponding to a *component* in [8] is represented using a rectangle, and each port in $\mathfrak{P}$ is represented using a circle, with lines connecting the components to the ports which they subsume. Fig. 3 shows the representation for a decomposition of the schema $\mathbf{E}_0$ of Sec. 1 (to be discussed in detail in the next section), and Fig. 2 illustrates this idea for the views $\{\Pi^{\mathbf{E}_3}_{AB}, \Pi^{\mathbf{E}_3}_{BC}, \Pi^{\mathbf{E}_3}_{CD}, \Pi^{\mathbf{E}_3}_{CE}\}$ of the relational schema $\mathbf{E}_3$ with the the single relation $R[ABCDE]$, constrained by the FDs in $\mathcal{F}_3 = \{A \rightarrow B, B \rightarrow C, C \rightarrow DE\}$. The set of ports is taken to be the three projections $\{\Pi^{\mathbf{E}_3}_{A}, \Pi^{\mathbf{E}_3}_{B}, \Pi^{\mathbf{E}_3}_{C}\}$.

In Fig. 2, the port $R[A]$ is shown with dashed lines because it is not necessary, as it is connected to only one component. Such a port is called *irrelevant*. More generally, returning to the general case of $\mathfrak{C}$ and $\mathfrak{P}$, call $\Gamma \in \mathfrak{P}$ a *port of* $\Gamma_i \in \mathfrak{C}$ if $\Gamma \sqsubseteq_{\mathbf{D}} \Gamma_i$, and call it an *essential port* of Γ_i if there is at least one other component Γ_j, distinct from Γ_i, for which it is also a port. Call $\Gamma \in \mathfrak{P}$ *relevant*

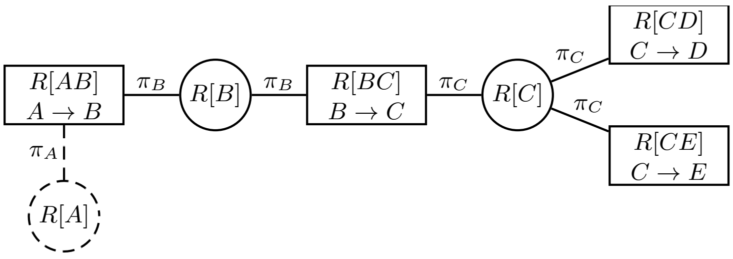


Fig. 2. The components of $\mathbf{E}_3$

for $\mathfrak{C}$ if it is an essential port for at least two distinct members of $\mathfrak{C}$, and call $\mathfrak{P}$ *completely relevant* for $\mathfrak{C}$ if every $\Gamma \in \mathfrak{P}$ is relevant for $\mathfrak{C}$. It is always possible to render $\mathfrak{P}$ completely relevant by removing elements which are not relevant. Let $\mathsf{RelRed}_{\mathfrak{C}}\langle\mathfrak{P}\rangle$ denote the subset of $\mathfrak{P}$ obtained by removing all elements which are not relevant for $\mathfrak{C}$.

Definition 2.4 (Independent updates). Using pairwise definability, it is possible to give a formal definition of what is meant by independent updates. First, let $u = \langle(N_1, N_1'), \ldots, (N_n, N_n')\rangle \in \mathsf{Updates}(\mathbf{V}_1) \times \ldots \times \mathsf{Updates}(\mathbf{V}_n)$. Call u *initial-state compatible* (resp. *final-state compatible*) for $\mathfrak{C}$ if $\langle N_1, \ldots, N_n\rangle$ (resp. $\langle N_1', \ldots, N_n'\rangle$) is compatible for $\mathfrak{C}$. If u is both initial-state and final-state compatible, it is called simply *compatible* for $\mathfrak{C}$. It is immediate that the set of all elements of $\mathsf{Updates}(\mathbf{V}_1) \times \ldots \times \mathsf{Updates}(\mathbf{V}_n)$ which are compatible for $\mathfrak{C}$ is in bijective correspondence with $\mathsf{Updates}(\mathbf{D})$. Of interest here, however, is the subset of the compatible updates which are independent. To this end, let $\mathfrak{u} \subseteq \mathsf{Updates}(\mathbf{V}_1) \times \ldots \times \mathsf{Updates}(\mathbf{V}_n)$ consist of compatible n-tuples. Call $\mathfrak{u}$ *independent* for $\mathfrak{C}$ if it is compatible for $\mathfrak{C}$ and the following two conditions are satisfied.

(ind1) For every $M \in \mathsf{LDB}(\mathbf{D})$, the corresponding identity n-tuple $\langle(\gamma_1(M), \gamma_1(M)), \ldots, (\gamma_n(M), \gamma_n(M))\rangle$ is in $\mathfrak{u}$ as well.

(ind2) For every n-element set $\{\langle(N_{i1}, N_{i1}'), \ldots, (N_{in}, N_{in}')\rangle \mid 1 \leq i \leq n\} \subseteq \mathfrak{u}$, the diagonal tuple $\langle(N_{11}, N_{11}'), \ldots, (N_{ii}, N_{ii}'), \ldots, (N_{nn}, N_{nn}')\rangle$ is in $\mathfrak{u}$ whenever it is initial-state compatible.

Condition (ind2) is the core of the definition, allowing one to "mix and match" updates from distinct n-tuples, subject only to the condition that the result is initial-state compatible. Condition (ind1) ensures that each view update may be considered alone, by matching it with identity updates from the other views.

For $\Gamma_i \in \mathfrak{C}$, define $\mathsf{UpdFam}\langle\Gamma_i, \mathfrak{P}\rangle$ to be the set of all updates on the schema $\mathbf{V}_i$ of Γ_i on which every $\Gamma \in \mathfrak{P}$ is constant and define $\mathsf{IndUpd}\langle\mathfrak{C}, \mathfrak{P}\rangle$ to be the set of initial-state compatible pairs in $\mathsf{UpdFam}\langle\Gamma_1, \mathfrak{P}\rangle \times \ldots \times \mathsf{UpdFam}\langle\Gamma_n, \mathfrak{P}\rangle$. The following result then provides the largest independent set.

Proposition 2.5. *Let $\mathfrak{C} = \{\Gamma_1, \ldots, \Gamma_n\}$ and $\mathfrak{P}$ be finite sets of views of $\mathbf{D}$, and suppose further that $\mathfrak{P}$ is completely relevant for $\mathfrak{C}$ and that $\mathfrak{C}$ is pairwise definable via $\mathfrak{P}$. Then for any $\mathfrak{u} \subseteq \mathsf{Updates}(\mathbf{V}_1) \times \ldots \times \mathsf{Updates}(\mathbf{V}_n)$ which is independent for $\mathfrak{C}$, $\mathfrak{u} \subseteq \mathsf{IndUpd}\langle\mathfrak{C}, \mathfrak{P}\rangle$.*

Proof. Let $\mathfrak{u} \subseteq \mathsf{Updates}(\mathbf{V}_1) \times \ldots \times \mathsf{Updates}(\mathbf{V}_n)$ be independent for $\mathfrak{C}$, and let $u = \langle(N_1, N_1'), \ldots, (N_n, N_n')\rangle \in \mathfrak{u}$. Choose $i \in \{1, \ldots, n\}$ and let $u' = \langle(N_1, N_1), \ldots, (N_{i-1}, N_{i-1}), (N_i, N_i'), (N_{i+1}, N_{i+1}), \ldots, (N_n, N_n)\rangle$. Thus, u' is obtained from u by retaining (N_i, N_i') and replacing each other entry with the identity which keeps it initial-state compatible. In view of the complete relevance of $\mathfrak{P}$, u' is constant on every view in $\mathfrak{P}$, since every such view is contained in at least two distinct members of $\mathfrak{C}$, at least one of which is held constant by the update. In particular, (N_i, N_i') is constant on all views $\Gamma \in \mathfrak{P}$ for which $\Gamma \sqsubseteq_{\mathbf{D}} \Gamma_i$. Since i was chosen arbitrarily, it follows that all of u must be constant on every view in $\mathfrak{P}$. □

Definition 2.6 (Compound components and external ports). The ideas of Definition 2.4 and Proposition 2.5 identify the conditions under which updates may be executed independently on the components in the set $\mathfrak{C}$ of data objects. However, not all updates are so representable. In particular, any update would change the state of of an essential port of $\mathfrak{P}$ is disallowed. Thus, this framework, by itself, is not adequate. When no atomic component in $\mathfrak{C}$ is adequate to support an update, the solution is to combine several components into one complex one. For example, referring to $\mathbf{E}_3$ of Definition 2.3 and Fig. 2, suppose that a transaction wishes to update the AB projection of $R[ABCDE]$. This is not possible within any single component; indeed, it requires an update on a port. The solution is to combine the components $R[AB]$ and $R[BC]$ into a single component defined by $R[ABC]$. The new set of components is then $\{\Pi^{\mathbf{E}_3}_{ABC}, \Pi^{\mathbf{E}_3}_{CE}, \Pi^{\mathbf{E}}_{CF}\}$, and an update to the AB-projection is now possible. The port $R[B]$ of this combination becomes *internal*, and hence irrelevant.

More generally, returning to the general case of $\mathfrak{C}$ and $\mathfrak{P}$ of Definition 2.3, a *compound component* over $\mathfrak{C}$ is any join of views in $\mathfrak{C}$. For $S \subseteq \mathfrak{C}$, the essential ports of the compound component $\bigsqcup C$ are exactly those $\Gamma \in \mathfrak{P}$ which are ports for some $\Gamma_i \in S$ as well as some $\Gamma_j \in \mathfrak{C} \setminus S$. Thus, for the join $\Pi^{\mathbf{E}_3}_{AB} \sqcup \Pi^{\mathbf{E}_3}_{BC}$, the only essential port is $R[C]$.

Discussion 2.7 (The support of independent updates). Continuing with the above framework, a transaction whose task is to perform an update must identify the components in $\mathfrak{C}$ which are necessary for the operations which it is to perform. The *claim set* $S \subseteq \mathfrak{C}$ which it is to hold must satisfy the following two conditions.

(u1) The update operations which it is to perform must be expressible within the view $\bigsqcup S$.

(u2) All essential ports of $\bigsqcup S$ must be constant under these update operations; that is, they are read(-only) claims but not write claims.

For a set of transactions to proceed independently, their claim sets may overlap only on their ports, and these overlaps identify the read claims. Any update is supportable by choosing S sufficiently large; indeed, by choosing $S = \mathfrak{C}$, all updates are possible (but without any parallelism).

It might seem a suitable strategy to allow a transaction to express its goal to update an arbitrary view Γ and then to seek a subset $S \subseteq \mathfrak{C}$ which "covers" Γ. However, this is not possible in general. Consider the example schema $\mathbf{E}_0$ of Sec. 1 and Figs. 1 and 3. Suppose that the view to be updated is the projection of Employee[Salary]. It makes no sense, in general, to insert salaries. They must be associated with employees. Thus, the transaction itself must have knowledge of the set $\mathfrak{C}$ of components and determine which ones to claim for its operations.

There is one further point which should be discussed briefly, and that is read claims. If a transaction claims an object $\bigsqcup S$, it may not need to be able to update all of it. For example, considering again the example $\mathbf{E}_0$, the task of a transaction may be to update employee salaries, based upon information about the department of each employee. Such a transaction would need to read claim the department information, but it would not require write access. The extension the model presented here to such read claims is straightforward, but due to space limitations it will not be developed further.

3 The Basic Components of a Relational Schema

The examples of Sec. 2 were all based upon classical "vertical" decomposition of relational schemata, which is not by itself adequate for defining useful read or write claims. In the context of the running schema $\mathbf{E}_0$, a transaction which is to update the salary of Alice should not need to claim the entire projection $\mathsf{Employee}[\mathsf{EmpID}, \mathsf{Salary}]$. Rather, it should suffice to claim just those tuples involving Alice. Thus, a complementary theory of *horizontal decomposition* is needed. That which is required for this work is very different from the horizontal decomposition of in [16, Ch. 5], which is based upon exceptions and *afunctional dependencies*. Unfortunately, that form of decomposition does not lead to pairwise definability. Rather, what is needed is an approach to horizontal decomposition which is based upon the relational operation of selection, just as vertical decomposition is based upon projection.

The context for this section is relational schema which are constrained by functional dependencies (FDs) and foreign-key dependencies (FKDs), which are undeniably the two most important types of constraints in real-world database schemata. Since classical vertical decomposition almost never considers inclusion dependencies (of which FKDs are a special case), it is prudent to begin with a short description of how they are incorporated into a vertical decomposition. To keep the focus on the key ideas, null values will not be considered; it will be assumed that all values are non null.

Discussion 3.1 (Conventions for vertical decomposition). The overall approach is to begin with a vertical decomposition, and then decompose each component into its horizontal sub-components. Thus, it is important to begin with a clarification of exactly what properties the vertical decomposition must have. First of all, the theory only applies to acyclic decompositions, which is equivalent to the existence of pairwise definable decompositions [2, Cond. 3.7]. Although that topic is usually approached from the perspective of join dependencies, cyclicity can also arise from decompositions arising entirely from FDs [1, Thm. 4]. Fortunately, such schemata occur rarely, if ever, in practice, but in any case, they are not covered by the theory presented here.

Figure 3 depicts the vertical decomposition for the schema $\mathbf{E}_3$, introduced in Sec. 1 and Fig. 1. For compactness, the relations $\mathsf{Employee}$, $\mathsf{Department}$, and Mentor are abbreviated to E, D, and M in the ports (the circles). $\underline{\text{Keys}}$ are underlined, while foreign keys have a wavy line beneath them. These abbreviations and conventions will also be used in that which follows.

In the classical theory of vertical decomposition, there is a tradeoff between 3NF, which always admits dependency-preserving decompositions but which may require non-key dependencies in some of the components, and BCNF, in which each component is governed only by key dependencies but for which

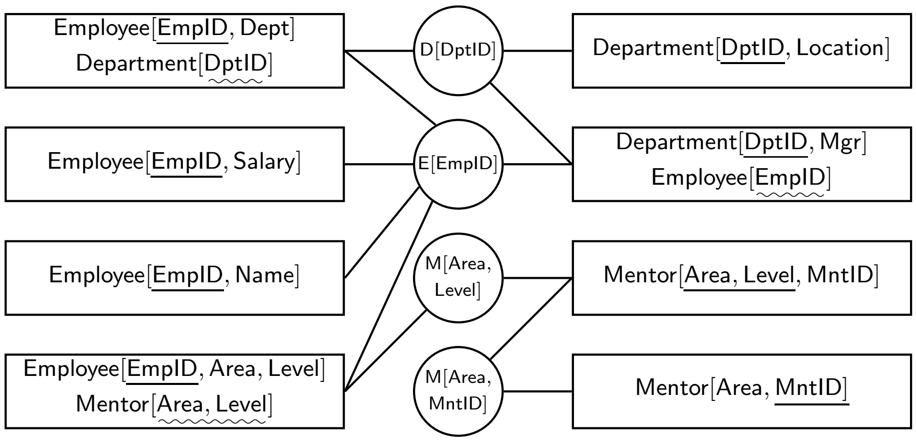


Fig. 3. The vertical components of the running example $\mathbf{E}_0$

dependency-preserving decomposition is not always possible. For this work, dependency preservation is essential. However, support for non-key dependencies within a component is also to be avoided. To address this dilemma, there is a trick which is arguably useless for classical normalization but which serves the purposes of support of independent updates very well. It is illustrated by the example of Mentor[MntID, Area, Level], governed by {Area, Level} $\rightarrow$ MntID and MntID $\rightarrow$ Area. The relation Mentor is "decomposed" into Mentor[MntID, Area, Level], in which only the key dependency is enforced, and Mentor[MntID, Area], in which only the local key dependency EmpID $\rightarrow$ Area is enforced. The latter FD is enforced in Mentor[MntID, Area, Level] via the common port M[Area, MntID]. This approach is taken also when a relation has more than one key, which must be split into several components, one for each key. For example, if the relation Department were to have an additional attribute DName which were also a (secondary) key, it would necessary to have two additional components, Department[DptID, DName] and Department[DptID, DName]. Their common port would be the entire relation Department[DptID, DName], but each key would be checked separately in its component. Thus, it is always possible to arrange things so that only one key dependency need be checked in each vertical component. In Fig. 3, the key to be checked in a given component is exactly that which is underlined.

In most cases each component relation may have only one non-key attribute. The only exception is foreign keys consisting of more than one attribute, which must be grouped. An example of the latter is Employee[EmpID, Area, Level]. Otherwise, for example, the relation Employee[EmpID, Salary, Dept] must be decomposed into Employee[EmpID, Salary] and Employee[EmpID, Dept], even though the composite is already in BCNF.

FKDs are of the form $R_1[F] \subseteq R_2[K]$, in which K is the (primary) key of R_2 and F is a set of attributes of R_1, called the *foreign key*. The case that F includes some key attributes of R_1 is not excluded. To preserve such a dependency in the component framework, both sides of the FKD must be contained in a single component. The convention that $R_2[K]$ be included in the component containing $R_1[F]$ is adopted. The satellite projection (e.g., $R_2[K]$) is then connected, via a port, to the corresponding component containing the full R_2 as the main relation. This is illustrated in three cases in Fig. 3.

Given a relational schema $\mathbf{D}$, a *simple key schema* is a set of projections of $\mathbf{D}$ with the following properties. First, it contains a *main relation* R, governed by a single key dependency; i.e., an FD which determines all other attributes. Second, it contains all projections onto their primary keys of the other relations R' in $\mathbf{D}$ for which the primary key of R' is a foreign key for R. It thus embodies the FKDs. All of the schemata in Fig. 3 are simple key schemata.

Convention 3.2 (The finite domain property). In that which follows for horizontal decomposition, it will always be assumed that each attribute A has the *finite domain property*; that is, the set $\mathsf{Dom}(A)$ of domain elements for A, the set of all possible values for attribute A, is finite. This condition is always met in real examples, and it simplifies the theory substantially by keeping the number of basic components in a horizontal decomposition finite.

Definition 3.3 (Views defined by selection). Just as projection is the defining operation for vertical decomposition, so too is selection the operation for horizontal decomposition. For a schema $\mathbf{D}$ whose relation symbols include $\{R_1, \ldots, R_k\}$, the notation $\sigma_\varphi\langle R_1, \ldots, R_k\rangle$ will be used to denote the selection φ applied to those relations. The *select view* $\Sigma_\varphi\langle R_1, \ldots, R_k\rangle$ has $\sigma_\varphi\langle R_1, \ldots, R_k\rangle$ as its underlying morphism.

The selects which will be applied to simple key schemata to obtain horizontal decompositions will always be of a particular form. The selection is defined only on the main relation, with all other relations "following" that select, based upon the embodied FKDs. The notation $\sigma_{(\varphi)^+}\langle R_1, \ldots, R_k\rangle$ will be used to denote such a selection, with $\Sigma_{(\varphi)^+}\langle R_1, \ldots, R_k\rangle$ the corresponding view. For example, considering the schema defined by the upper-left rectangle of Fig. 3, $\Sigma_{(\mathsf{EmpID}=\mathrm{Alice})^+}\langle \mathsf{E}[\underline{\mathsf{EmpID}}, \mathsf{Dept}], \mathsf{D}[\utilde{\mathsf{DptID}}]\rangle$ has selection morphism $\sigma_{((\mathsf{EmpID}=\mathrm{Alice}))^+}\langle \mathsf{E}[\underline{\mathsf{EmpID}}, \mathsf{Dept}], \mathsf{D}[\utilde{\mathsf{DptID}}]\rangle$ which is the same as the selection $\sigma_{(\mathsf{EmpID}=\mathrm{Alice})\wedge(\mathsf{Dept}=\mathsf{DptID})}\langle \mathsf{E}[\underline{\mathsf{EmpID}}, \mathsf{Dept}], \mathsf{D}[\utilde{\mathsf{DptID}}]\rangle$, while the selection $\sigma_{((\mathsf{EmpID}=\mathrm{Alice})\wedge(\mathsf{Dept}=\mathrm{Research}))^+}\langle \mathsf{E}[\underline{\mathsf{EmpID}}, \mathsf{Dept}], \mathsf{D}[\utilde{\mathsf{DptID}}]\rangle$ is identical to $\sigma_{(\mathsf{EmpID}=\mathrm{Alice})\wedge(\mathsf{Dept}=\mathrm{Research})\wedge(\mathsf{Dept}=\mathsf{DptID})}\langle \mathsf{E}[\underline{\mathsf{EmpID}}, \mathsf{Dept}], \mathsf{D}[\utilde{\mathsf{DptID}}]\rangle$.

If $\mathbf{D}$ is a simple key schema, *simple key select* on $\mathbf{E}$ is a select view $\Sigma_{(K=t)^+}\langle \mathbf{E}\rangle$, where $K = A_1 \ldots A_k$ is the key of the main relation of $\mathbf{D}$ and $t = (\mathrm{a}_1, \ldots, \mathrm{a}_k) \in \mathsf{Dom}(A_1) \times \ldots \times \mathsf{Dom}(A_k)$. Thus, in a simple key select, only a selection on the primary key of the main relation is allowed, with that selection extending to those parts of foreign keys which reference the primary key. For example, $\Sigma_{(\mathsf{EmpID}=\mathrm{Alice})^+}\langle(\mathsf{E}[\underline{\mathsf{EmpID}}, \mathsf{Dept}], \mathsf{D}[\utilde{\mathsf{DptID}}])\rangle$ is a simple key select while $\Sigma_{((\mathsf{EmpID}=\mathrm{Alice})\wedge(\mathsf{Dept}=\mathrm{Research}))^+}\langle(\mathsf{E}[\underline{\mathsf{EmpID}}, \mathsf{Dept}], \mathsf{D}[\utilde{\mathsf{DptID}}])\rangle$ is not.

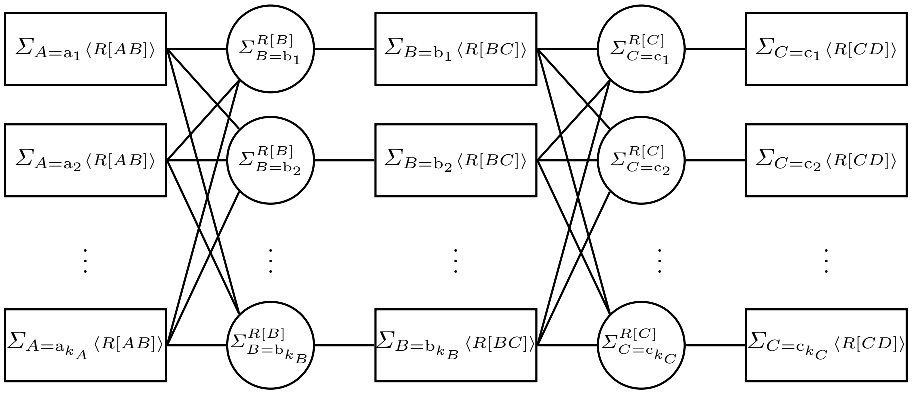
Simple key schemata may be decomposed on a key-by-key basis into simple key selects, with complete independence. The following observation, whose proof is immediate, recaptures this.

Observation 3.4 (Horizontal decomposition of a simple key schema). *Let* $\mathbf{D}$ *be a simple key schema with primary key* $K = A_1 \ldots A_k$*, and whose relations include* $\{R_1, \ldots, R_k\}$*. Then* $\{\Sigma_{(K=S)^+}\langle R_1, \ldots, R_k\rangle \mid S \in \mathsf{Dom}(A_1) \times \ldots \times \mathsf{Dom}(A_k)\}$ *is pairwise definable, with the ports the relativized zero views.* □

Examples 3.5 (Combined horizontal and vertical decomposition). To visualize the nature of decomposition using simple key selects as the horizontal part of a vertical-horizontal decomposition, it is best to begin with a schema which is simpler and more regular than that running example $\mathbf{E}_0$. To that end, let $\mathbf{E}_2$ be as defined in Definition 2.3, having a vertical decomposition into the basic schema components $\{\Pi^{\mathbf{E}_2}_{AB}, \Pi^{\mathbf{E}_2}_{BC}, \Pi^{\mathbf{E}_2}_{CD}\}$, The combined horizontal and vertical decomposition is depicted in Fig. 4. Each horizontal decomposition is actually represented by a vertical stack of components defined by simple key selects. Note that the ports are each selections on single elements. The horizontal components associated with $R[AB]$ and $R[BC]$ have connections to each projection on the non-key attribute.

Now return to the running example $\mathbf{E}_0$ of Fig. 3, the vertical component defined by $\mathsf{Employee}[\underline{\mathsf{EmplD}}, \mathsf{Salary}]$. There is one horizontal component for each element of $\mathsf{Dom}(\underline{\mathsf{EmplD}})$. To update the salary of Alice, only the component $\Sigma_{(\mathsf{EmplD}=\text{Alice})^+}\langle \mathsf{E}[\underline{\mathsf{EmplD}}, \mathsf{Salary}]\rangle$ need be claimed; the salary of any other employee may be updated independently.

Next consider changing the department of Alice. The relevant component is $\Sigma_{((\mathsf{EmplD}=\text{Alice}))^+}\langle \mathsf{E}[\mathsf{EmplD}, \mathsf{Dept}], \mathsf{D}[\mathsf{DptID}]\rangle$ Again, only the tuple corresponding to Alice need be claimed; all others are available to other transactions. However, the entire set of names of departments is read claimed, since $\Sigma_{((\mathsf{EmplD}=\text{Alice}))^+}\langle \mathsf{E}[\mathsf{EmplD}, \mathsf{Dept}], \mathsf{D}[\mathsf{DptID}]\rangle$ has a port connecting to each



$\Sigma_{A=\mathrm{a}_1}\langle R[AB]\rangle$
$\Sigma_{A=\mathrm{a}_2}\langle R[AB]\rangle$
$\Sigma_{A=\mathrm{a}_{k_A}}\langle R[AB]\rangle$
$\Sigma^{R[B]}_{B=\mathrm{b}_1}$
$\Sigma^{R[B]}_{B=\mathrm{b}_2}$
$\Sigma^{R[B]}_{B=\mathrm{b}_{k_B}}$
$\Sigma_{B=\mathrm{b}_1}\langle R[BC]\rangle$
$\Sigma_{B=\mathrm{b}_2}\langle R[BC]\rangle$
$\Sigma_{B=\mathrm{b}_{k_B}}\langle R[BC]\rangle$
$\Sigma^{R[C]}_{C=\mathrm{c}_1}$
$\Sigma^{R[C]}_{C=\mathrm{c}_2}$
$\Sigma^{R[C]}_{C=\mathrm{c}_{k_C}}$
$\Sigma_{C=\mathrm{c}_1}\langle R[CD]\rangle$
$\Sigma_{C=\mathrm{c}_2}\langle R[CD]\rangle$
$\Sigma_{C=\mathrm{c}_{k_C}}\langle R[CD]\rangle$

Fig. 4. Visualization of the horizontal and vertical components for $\mathbf{E}_6$

horizontal component of $\mathsf{Department}[\underline{\mathsf{DptID}}, \mathsf{Location}]$. Thus, while it is possible to execute other updates concurrently which read the list of department names, it is not possible for any transaction to update this list. More will be said about this point in Discussion 3.6.

Finally, suppose that a transaction wishes to add $(\mathrm{French}, 1, \mathrm{DuBois})$ and $(\mathrm{German}, 1, \mathrm{Dimpflmeier})$ to $\mathsf{Mentor}[\underline{\mathsf{Area}, \mathsf{Level}}, \mathsf{MntID}]$. The constraint $\mathsf{MntID} \to \mathsf{Area}$ must be verified to support this change. This update may be effected with support from a write claim for $\Sigma_{((\mathsf{Area} \in \{\mathrm{French}, \mathrm{German}\}) \wedge (\mathsf{Level}=1))^+} \langle \mathsf{Mentor}[\underline{\mathsf{Area}, \mathsf{Level}}, \mathsf{MntID}] \rangle$, which is the join $\Sigma_{((\mathsf{Area}=\mathrm{French}) \wedge (\mathsf{Level}=1))^+} \langle \mathsf{Mentor}[\underline{\mathsf{Area}, \mathsf{Level}}, \mathsf{MntID}] \rangle$

$$\sqcup\, \Sigma_{((\mathsf{Area}=\mathrm{German}) \wedge (\mathsf{Level}=1))^+} \langle \mathsf{Mentor}[\underline{\mathsf{Area}, \mathsf{Level}}, \mathsf{MntID}] \rangle$$

of two simple key selects. It also effects a read claim on the port defined by $\mathsf{Mentor}[\mathsf{Area}, \underline{\mathsf{MntID}}]$. The transaction read claims this port, and hence the entire component of the same name. If DuBois is a already a mentor for French, and Dimpflmeier is already a mentor for German (as recorded in $\mathsf{Mentor}[\mathsf{Area}, \underline{\mathsf{MntID}}]$), then no further claims are necessary. However, if this is not the case, then $\mathsf{Mentor}[\mathsf{Area}, \underline{\mathsf{MntID}}]$ must be write claimed for DuBois and Dimpflmeier as well. Since this read claims all Area values in $\mathsf{Mentor}[\mathsf{Area}, \underline{\mathsf{MntID}}]$, this effectively claims the entire Mentor relation, and no parallel updates are possible (although much may still be read).

Discussion 3.6 (Range restriction). As illustrated in Examples 3.5, claiming a relation even for one key value claims every non-key value. For example, for a transaction T_1 to place new employee Alice in the Research department, the view $\Sigma_{(\mathsf{EmpID}=\mathrm{Alice})^+} \langle \mathsf{E}[\mathsf{EmpID}, \mathsf{Dept}], \mathsf{D}[\mathsf{DptID}] \rangle$ must be claimed, which reserves every possible value for Dept as a new value for the tuple with key Alice. Thus, a second transaction T_2, whose task it is to delete the Education department, could not proceed independently because it is not known that T_1 will not change the department for Alice to Education. It might seem that a solution would be for T_1 to claim only $\Sigma_{((\mathsf{EmpID}=\mathrm{Alice}) \wedge (\mathsf{Dept}=\mathrm{Research}))^+} \langle \mathsf{E}[\underline{\mathsf{EmpID}}, \mathsf{Dept}], \mathsf{D}[\utilde{\mathsf{DptID}}] \rangle$ However, that would open the door for another transaction to put Alice into another department, say TechSupport, concurrently, since it is impossible to verify within $\Sigma_{((\mathsf{EmpID}=\mathrm{Alice}) \wedge (\mathsf{Dept}=\mathrm{Research}))^+} \langle \mathsf{E}[\underline{\mathsf{EmpID}}, \mathsf{Dept}], \mathsf{D}[\utilde{\mathsf{DptID}}] \rangle$ whether the FD $\mathsf{EmpID} \to \mathsf{Dept}$ is satisfied for $\mathsf{EmpID} = \mathrm{Alice}$. Since the whole purpose of this approach is to detect and prevent such conflicts and support truly independent updates, simply ignoring this conflict is not acceptable. Fortunately, there is a solution, provided the transaction provides a bit more information. Roughly, the idea is that the transaction claims the data object $\Sigma_{((\mathsf{EmpID}=\mathrm{Alice}) \wedge (\mathsf{Dept}=\mathrm{Research}))^+} \langle \mathsf{E}[\underline{\mathsf{EmpID}}, \mathsf{Dept}], \mathsf{D}[\utilde{\mathsf{DptID}}] \rangle$ as suggested, and for the life of that claim, the system excludes any other possibilities for tuples with $\mathsf{EmpID} = \mathrm{Alice}$. This does not reduce possible parallelism in any way, since only one transaction may have write privileges on a field of a tuple with a given key. Furthermore, it allows the transaction which is to delete the Education department to proceed independently (provided that no other employee works in that department). This approach may be used to increase the possible concurrency for the examples involving the Mentor at the end of Examples 3.5 as well.

It is a straightforward exercise to extend the framework of Sec. 2 to incorporate these additional features, called *range restriction.* Unfortunately, space limitation preclude a full presentation here.

4 Conclusions and Further Directions

A theory of updateable data objects has been presented. A key of this approach is that transactions may claim data objects which overlap on read areas. It requires more structural analysis of the schema, but in return it supports a finer grain of concurrency than in achievable with traditional models. It is particularly suited to interactive applications, which can tolerate more preprocessing for a transaction to begin but are much more sensitive to long waits and aborts.

Important further directions include the following.

Application to cooperative update: An approach to the support of cooperative updates which is based upon components has been developed recently [11], [10]. Indeed, the ideas for this paper began as a search for appropriate ways to support concurrency in such an environment, and the topic will be developed further.

Application to nondeterministic update requests: A key aspect of the approach to cooperative updates in [11] and [10] is that requests may be *nondeterministic*; that is, a user may request that one of a number of alternative updates be supported. For example, a business travel request may involve alternatives regarding hotel, flight, date, *etc.*. Some of these alternatives may become impossible as the processing of the transaction proceeds. The model of conflict developed here seems well suited to identifying and eliminating parts of the nondeterministic request which conflict with other needs and thus cannot be supported, while allowing the others to proceed.

Acknowledgement. Some of the ideas reported here have their origins in discussions with Peggy Schmidt. Her contribution is gratefully acknowledged.

References

1. Aho, A.V., Beeri, C., Ullman, J.D.: The theory of joins in relational databases. ACM TODS 4(3), 297–314 (1979)
2. Beeri, C., Fagin, R., Maier, D., Yannakakis, M.: On the desirability of acyclic database schemes. JACM 30(3), 479–513 (1983)
3. Bernstein, P.A., Hadzilacos, V., Goodman, N.: Concurrency Control and Recovery in Database Systems. Addison-Wesley, Reading (1987)
4. Elmasri, R., Navathe, S.B.: Fundamentals of Database Systems, 5th edn. Addison Wesley, Reading (2006)

5. Eswaran, K.P., Gray, J., Lorie, R.A., Traiger, I.L.: The notions of consistency and predicate locks in a database system. ACM Comm. 19(11), 624–633 (1976)
6. Hegner, S.J.: An order-based theory of updates for closed database views. Ann. Math. Art. Intell. 40, 63–125 (2004)
7. Hegner, S.J.: The complexity of embedded axiomatization for a class of closed database views. Ann. Math. Art. Intell. 46, 38–97 (2006)
8. Hegner, S.J.: A model of database components and their interconnection based upon communicating views. In: Jakkola, H., Kiyoki, Y., Tokuda, T. (eds.) Information Modelling and Knowledge Systems XIX. Frontiers in Artificial Intelligence and Applications, pp. 79–100. IOS Press, Amsterdam (2008)
9. Hegner, S.J.: Semantic bijectivity and the uniqueness of constant-complement updates in the relatiional context. In: Schewe, K.-D., Thalheim, B. (eds.) SDKB 2008. LNCS, vol. 4925, pp. 172–191. Springer, Heidelberg (2008)
10. Hegner, S.J.: A simple model of negotiation for cooperative updates on database schema components. In: Kiyoki, Y., Tokuda, T., Heimbürger, A., Jaakkola, H., Yoshida, N. (eds.) Frontiers in Artificial Intelligence and Applications XX11 (in Press, 2011)
11. Hegner, S.J., Schmidt, P.: Update support for database views via cooperation. In: Ioannidis, Y., Novikov, B., Rachev, B. (eds.) ADBIS 2007. LNCS, vol. 4690, pp. 98–113. Springer, Heidelberg (2007)
12. Herlihy, M., Weihl, W.E.: Hybrid concurrency control for abstract data types. J. Comput. System Sci. 43(1), 25–61 (1991)
13. Kemper, A., Eickler, A.: Datenbanksysteme, p. 6. Oldenbourg, Auflage (2006)
14. Klug, A.C.: Locking expressions for increased database concurrency. J. Assoc. Comp. Mach. 30(1), 36–54 (1983)
15. Papadimitriou, C.: The Theory of Database Concurrency Control. Computer Science Press, Rockville (1986)
16. Paredaens, J., De Bra, P., Gyssens, M., Van Gucht, D.: The Structure of the Relational Database Model. Springer, Heidelberg (1989)
17. Sampaio, M.C., Turc, S.: Cooperative transactions: A data-driven approach. In: 29th Annual Hawaii International Conference on System Sciences (HICSS-29), Maui, Hawaii, January 3-6, pp. 41–50. IEEE Computer Society, Los Alamitos (1996)
18. Wieczerzycki, W.: Transaction management in databases supporting collaborative applications. In: Litwin, W., Morzy, T., Vossen, G. (eds.) ADBIS 1998. LNCS, vol. 1475, pp. 107–118. Springer, Heidelberg (1998)

Using a Time Granularity Table for Gradual Granular Data Aggregation

Nadeem Iftikhar and Torben Bach Pedersen

Aalborg University, Department of Computer Science, Selma Lagerløfs Vej 300,
9220 Aalborg Ø, Denmark
{nadeem,tbp}@cs.aau.dk

Abstract. The majority of today's systems increasingly require sophisticated data management as they need to store and to query large amounts of data for analysis and reporting purposes. In order to keep more "detailed" data available for longer periods, "old" data has to be reduced gradually to save space and improve query performance, especially on resource-constrained systems with limited storage and query processing capabilities. A number of data reduction solutions have been developed, however an effective solution particularly based on gradual data reduction is missing. This paper presents an effective solution for data reduction based on gradual granular data aggregation. With the gradual granular data aggregation mechanism, older data can be made coarse-grained while keeping the newest data fine-grained. For instance, when data is 3 months old aggregate to 1 minute level from 1 second level, when data is 6 months old aggregate to 2 minutes level from 1 minute level and so on. The proposed solution introduces a time granularity based data structure, namely a relational time granularity table that enables long term storage of old data by maintaining it at different levels of granularity and effective query processing due to a reduction in data volume. In addition, the paper describes the implementation strategy derived from a farming case study using standard technologies.

Keywords: Data reduction, data aggregation, gradual granular data aggregation, multi-granular data.

1 Introduction

In order to have both fine grained data and to store data for as long time periods as possible, we need to aggregate the data in an intelligent way. Further, the queries that are being used by various applications for extracting the logged data must continue to work and generate valid results even after aggregation. Furthermore, as the detailed data grows older, it slowly loses its value or may not have the same value as before. Examples include details about spray related tasks in the farming business, such as *task start time*, *task end time*, *task duration*, *task status, spraying pressure, number of active nozzles, working width of the nozzles etc*.

Although these details were useful for some time, after a year or so they may not be of much importance and they could reduce the query performance. Therefore, to save disk space and to perform efficient query processing there may be two options of data reduction, either to delete older data or to aggregate it. However, the major problem with deleting the older data could be organizational or governmental level data retention laws; therefore it may not be a feasible solution. Instead, data aggregation is preferable. Furthermore, the aggregated data could be quite useful for analysis purposes. For example, by using spray related aggregated data, maintenance of the farming machinery such as when the service of the spraying equipment is due or to determine the working life of different components of the equipment as well as future planning on task management could be done.

This paper presents an effective data reduction solution based on gradual granular data aggregation and time granularity table. The solution saves vital storage capacity and keeps data for long time periods. The gradual aggregation solution for data reduction is mainly based on the time granularity table and derived from the specifications provided by [1]. To the best of our knowledge, this paper is the first to present and demonstrate a data reduction solution that maintains the data at different levels of granularity.

The paper is structured as follows. Section 2 presents the real-world farming case study and explains the motivation behind the proposed gradual granular data aggregation. Section 3 presents the handling of time granularities. Section 4 describes the data aggregation methods. Section 5 evaluates the proposed approach. Section 6 presents the related work. Finally, Section 7 summarizes and points to the future research.

2 Motivation

This section presents a real-world case study based on the farming business. The case study is a result of LandIT [2] that was an industrial collaboration project about developing technologies for integration, aggregation and exchange of data between embedded farming devices and other farming-related IT systems, both for operational and business intelligence purposes. The farming devices represent computing devices with the ability to produce and store data for instance, spray control devices, climate control and production monitoring devices and so on. These devices are either installed at a fixed location of a farm, or on a moving object such as a tractor.

One of the main goals of this case study is to aggregate data at different levels of granularity, according to the needs of the application domain. We use the case study to illustrate the kind of challenges faced by aggregating data at different levels of granularity, which are addressed by this paper.

This case study concerns spray related data in a field that has to be logged in order to comply with environmental regulations. The requirement analysis of the case study result in two main types of data: *sensor data* and *setting data*. The sensor data set contains the following attributes: tractor speed in kilometers/hour, application rate in liters/hectare, distance covered in meters, area sprayed in hectares, time in seconds, volume of chemical used in liters etc. The setting data set contains the following attributes: nozzle type, nozzle size, nozzle flow rate, number of active nozzles, working width of the nozzles, spraying pressure etc. Further, the logged data is initially kept in detailed format in the *Datalog* table that consists of following attributes: *Taskid, Timeid, DatalogDDI* and *Datalogvalue*. The Taskid represents activities to distinguish all the work that is carried out by a contractor for a farmer in a particular field of a farm. The Timeid specifies a recording of a time event at different levels of granularity. The DatalogDDI represents the codes (DDIs) against which actual values are being logged from different devices and the Datalogvalue specifies a numeric value against a specific DDI or code. For example, a single instance of the Datalog table looks as follows: a Datalogvalue of 165 is recorded against DatalogDDI “0074” that stands for “total area sprayed in hectares” at a specific time for a specific task. Furthermore, in order to save storage space, LandIT data requirements also impose that older data should not be deleted; however, it should be reduced gradually by aggregating it *per task level*.

The readings from the devices are logged and then aggregated after some interval in time, or if lack of space occurs. In addition, data values are aggregated differently. Some of the values have to be aggregated using SUM() or MAX() and some using COUNT(). For example, total distance travelled by the tractor should be aggregated using SUM(), total number of tasks should be aggregated using COUNT() and so on. Moreover, according to the requirements the aggregation should not be a onetime process; rather it should be a continuous process, meaning that data should be aggregated gradually.

The main reason behind aggregating data gradually is to maintain data at different levels of granularity ranging from fine-grained to coarse-grained data, where each level can be used for analysis and reporting purposes. For example as shown in Fig. 1, if we initially have the granularity of spray related data at a second level (meaning that each data value is being recorded every second) then it could be aggregated from a second level to a minute level if it is more than three months old, further to a 2 minutes level if it is more than six months old, furthermore to a 10 minutes level if it is more than twelve months old and so on. In conclusion, an effective data reduction based on gradual granular aggregation is important not only for the farming industry but for any other type of industry in which significant amounts of data are generated.

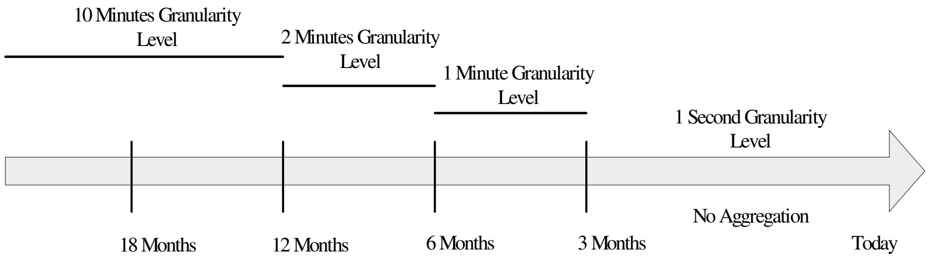

10 Minutes Granularity Level
2 Minutes Granularity Level
1 Minute Granularity Level
1 Second Granularity Level
No Aggregation
18 Months
12 Months
6 Months
3 Months
Today

Fig. 1. Data at 10 minutes, 2 minutes, minute and second granularity levels

3 Handling Time Granularities

In this section, we first describe the structure of the time granularity table in order to store multi-granular data, followed by an example based on the real-world case study presented in Section 2. The time granularity table is used along with the Datalog table (Section 2) to store data at different levels of granularity. The Timegranularity table presented in this paper is based on a single hierarchy that is further composed of numerous one-to-many relationships. Moreover, the proposed time granularity based solution can easily be applied to the fact tables having time as a dimension, in order to specify instances of fact table at different levels of granularity.

3.1 The Timegranularity Table

In the case study presented in Section 2, it is preferred not to throw away the older data; instead it should be kept in a summarized form. Again, it is not a good option to summarize the older data at the same granularity level; instead it should be summarized at different levels of granularity. This means that newer data which is needed most should be fine-grained, and older data which may not be needed as frequently as the newer data but could be used for analysis purposes, should be coarse-grained. This paper presents an effective technique based on the semantic foundation provided by [1] for gradual granular data aggregation.

The implementation of this technique is done in the form of a time granularity based data structure, namely a relational Timegranularity table, presented as Table 1. In the proposed table, in addition to the standard time associated attributes, a new attribute *Gran* (*Granularity)* is added, in order to handle granularity at different levels rather than at a single level. The attribute Granularity represents the level of detail of each time instance stored in the Timegranularity table. Thus, with the inclusion of this new attribute in the time dimension table, the associated Datalog or fact table is no longer restricted to store data only at the same single level of granularity. Furthermore, the reason to consider the time granularity only, rather than any other granularity(s), is that different levels of granularity always occur for time and the solution also works for other granularity phenomena.

Table 1. Snapshot of the Timegranularity table

Tid	Y	Q	M	D	POD	4H	H	20M	10M	2M	M	S	Gran
1	2009	Null	Null	Null	Null	Null	Null	Null	Null	Null	Null	Null	Y
2	2009	3	Null	Null	Null	Null	Null	Null	Null	Null	Null	Null	Q
3	2009	3	8	Null	Null	Null	Null	Null	Null	Null	Null	Null	M
4	2009	3	8	12	Null	Null	Null	Null	Null	Null	Null	Null	D
5	2009	3	8	12	Day[8-16[	Null	Null	Null	Null	Null	Null	Null	POD
6	2009	3	8	12	Day[8-16[	[12-16[	Null	Null	Null	Null	Null	Null	4H
7	2009	3	8	12	Day[8-16[	[12-16[	15	Null	Null	Null	Null	Null	H
8	2009	3	8	12	Day[8-16[	[12-16[	15	[20-40[	Null	Null	Null	Null	20M
9	2009	3	8	12	Day[8-16[	[12-16[	15	[20-40[	[20-30[	Null	Null	Null	10M
10	2009	3	8	12	Day[8-16[	[12-16[	15	[20-40[	[20-30[	[24-26[	Null	Null	2M
11	2009	3	8	12	Day[8-16[	[12-16[	15	[20-40[	[20-30[	[24-26[	25	Null	M
12	2009	3	8	12	Day[8-16[	[12-16[	15	[20-40[	[20-30[	[24-26[	25	40	S

The Timegranularity table contains the following attributes: *Tid (Timeid),* which is an auto generated primary key attribute. *Y (Year), Q (Quarter), M (Month)* and *D (Day)*, represent the standard calendar year, quarter, month and day. The values of these attributes depend on the level of granularity. For example, row number 1 in the Timegranularity table has a granularity at the year level and it is read as follows: year = 2009. Since the granularity of this row is at the year level, therefore the values for quarter, month, day, part of day, four-hour, hour, twenty-minute, ten-minute, two-minutes, minute and second are NULL. Row number 4 has a granularity at the day level and it is read as follows: year = 2009, 3rd quarter, month of August and day in day = 12. As the granularity level of this row is at day level; therefore the values of rest of the attributes are NULL. Further, *POD (Partofday)* represents an eight-hour time span. Hence each day, is divided into three eight-hour time spans, from 0000 to 0800 hours, 0800 to 1600 hours and 1600 to 0000 hours. The symbol shown in the Timegranularity table for Partofday is Day[8-16[, which means that it is the second time span of the day, where 0800 hours are included but 1600 hours are excluded. 1600 hours will be included in the next time span. Furthermore, *4H (4Hour)* represents a four-hour time span. *H (Hour)* represents the standard hour in 24 hours time format. *20M (20Minute), 10M (10Minute)* and *2M (2Minute),* represents twenty-minute, ten-minute and two-minute time spans. Moreover, *M (Minute)* and *S (Second)* represent the standard minute and second in any time format. Finally, *Gran (Granularity)* represents the level of granularity of each row. For instance, row number 12 has a granularity at the second level and it is read as follows: year = 2009, 3rd quarter, month of August, day in month = 12, partofday = 0800 – 1600 (eight-hour time span), four-hour = 1200 – 1600 (four-hour time span), hour = 15, twenty-minute = 20 – 40 (twenty-minute time span), ten-minute = 20 – 30 (ten-minute time span), two-minute = 24 – 26 (two-minute time span), minute = 25 and second = 40. Lastly, the granularity column represents the time granularity of each stored time instance. To make sense of this flexible structure, consider the Datalog table (Section 2). For instance if the values of tractor speed are logged at second granularity level, then they could be aggregated at the minute level, 2 minutes level and so on. This whole process could be done gradually, in other words, the data log values could point to time periods of any predefined granularity.

3.2 Gradual Granular Data Aggregation Example

As an example, we have considered the data consists of 1,800,000 rows (data log values) distributed uniformly across a two-year time period; values were logged for 30 DDIs in total, out of which 20 DDIs at a granularity of one second per 20 values with 40,000 readings for each DDI and 10 DDIs at a granularity of one minute per 10 values with 20,000 readings for each DDI . First, the first level aggregation is applied (using the time granularity table); "*for data which is more than three months old, aggregate to the minute granularity level from the second granularity level*". The data is aggregated at *per task level.* After applying the first level aggregation, the number of rows is reduced from 1,800,000 to 767,500 rows, as shown in Fig. 2a. Out of the 767,500 rows, 542,500 rows are at the minute granularity level and 225,000 rows are at the second granularity level, since they are less than three months old.

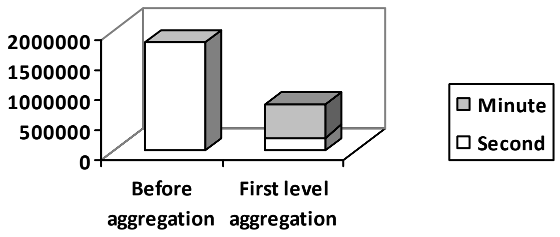


Fig. 2a. Data reduction after first level aggregation (note zoom on y-axis)

Second, the second level aggregation is applied "*for data which is more than six months old, aggregate to the 2 minutes granularity level from the minute granularity level*". After applying the second level aggregation, the number of rows is reduced from 767,500 to 586,666 rows, as shown in Fig. 2b. Out of the 586,666 rows, 191,666 rows are at the 2 minutes granularity level, 170,000 rows are at the minute granularity level, since they are less than six months old and 225,000 rows are at the second granularity level. The reduction in the number of rows at this level is about 24 %, compared to the first level aggregation, where the reduction was about 57 %. The reason is that the first level aggregation was applied to a larger set of fine-grained data, whereas the second level aggregation is applied to a smaller set of coarse-grained data.

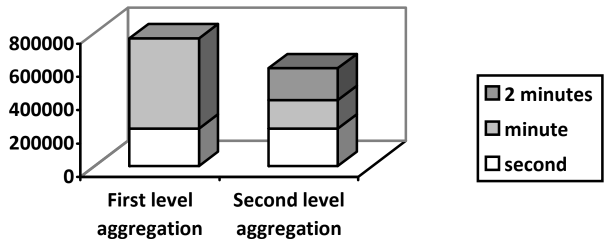


Fig. 2b. Data reduction after second level aggregation (note zoom on y-axis)

Last, the third level aggregation is applied. "*for data which is more than twelve months old, aggregate to the 10 minutes granularity level from the 2 minutes granularity level*". After applying the third level aggregation, the number of rows is reduced from 586,666 to 484,443 rows that is further 18 % reduction in the number of rows, as shown in Fig. 2c. Out of these 484,443 rows, 25,555 rows are at the 10 minutes granularity level, 63,888 rows are at the 2 minutes granularity level since they are less than twelve months old, 170,000 rows are at the minute granularity level and 225,000 rows are at the second granularity level.

In total, approximately 73 % reduction in the number of rows has been achieved. Similarly, the graph in Fig. 3 shows the effects on an approximate set of data log values for a five-year time period. The graph shows the results in the form of "before aggregation" at the second level and "after aggregation" to the minute level after three months,

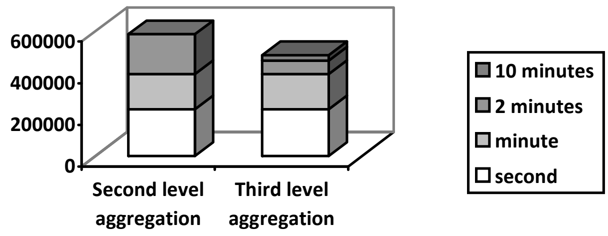


Fig. 2c. Data reduction after third level aggregation (note zoom on y-axis)

the 2 minutes level after six months and the 10 minutes level after twelve months. Based on the same scenario, if we assume that, on average only one task is done each day in a five year time period, the Datalog table, without aggregation could end up with approximately two hundred million rows. If gradual granular aggregation is applied then this number is reduced from two hundred million to approximately ten million six hundred thousand, this is an approximately 95 % reduction in the number of rows.

Although, the reduction in the number of rows is quite large however, it really depends on the specific data application, since none of the data is deleted and data exists at different levels of granularity ranging from fine-grained to coarse-grained for analysis and reporting purpose, as seen in Fig.3. It shows the data for the five years time period at four different levels. Level one “white” is quite visible that contains data at the second granularity level this means that the data is not aggregated at all because it is less than three months old; accordingly it has to be kept as fine-grained. Furthermore, higher levels in Fig. 3 shows the data that is aggregated at three levels: minute, 2 minutes and 10 minutes. The minute level shows the data that is more than three months old. Further, the 2 minutes level shows the data that is more than six months old. Furthermore, the 10 minutes level shows the data that is more than twelve months old.

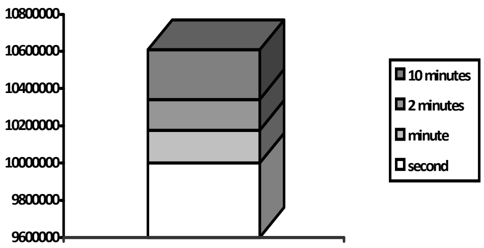


Fig. 3. Row counts for five years’ data (note zoom on y-axis)

4 Aggregation Methods

The proposed methods aggregate the data log values *per task level*. In the requirement specifications of the case study presented in Section 2, the launch of the data aggregation process should be based on a time interval or lack of storage space. For time interval based data aggregation, the aggregation methods would be called manually. For example, a call to *aggregation_at_minute_level* method would aggregate the data that is older than three months at the minute level from the second granularity level. Similarly, a call to *aggregation_at_two_minute_level* method would aggregate the data that is older than six months at the 2 minutes level from the minute granularity level and so on.

Further, for lack of space based data aggregation, the aggregation methods could be executed automatically with the help of a trigger. The trigger is based on a row count mechanism and it would be fired when the number of rows exceeded a threshold value. For example, if the threshold value reaches one million then the rows which are older than three months are aggregated to the minute granularity level from the second granularity level. Similarly, when the rows reaches one million level again then once more the rows which are less than three months old are not going to be aggregated at all, only the rows which are older than three months and not aggregated at the minute granularity level before, would be aggregated at the minute level.

The complete aggregation process consists of 1) aggregating the existing rows based on single or different levels of granularity, 2) generating the new rows in the time granularity table in order to point to the higher granularity rows in the Datalog table, 3) inserting the newly aggregated rows in the Datalog table and 4) deleting the previous rows from the Datalog table. There are numerous methods that work together to perform the desired gradual granular data aggregation, however in this paper, we have presented the methods that aggregate the data at the minute granularity level.

```
CREATE PROCEDURE aggregation_at_minute_level()
MODIFIES SQL DATA
BEGIN

  DECLARE total_task INT;
  DECLARE task_counter INT;
  DECLARE taskid INT;
  SET total_task= count_task();
  SET task_counter = 1;
     WHILE taskcounter <= totaltask DO
        SET taskid = get_task_id();
        CALL aggregate_data(taskid);
        CALL delete_detailed_data(taskid);
        SET task_counter =task_counter +1;
     END WHILE;

END
```

The *aggregation_at_minute_level* method aggregates the data from a lower granularity level (lower than a minute) to the minute granularity level and it works as follows. Other methods, with almost similar configuration, aggregate data at a higher granularity level for example 2 minutes, 10 minutes, 20 minutes and so on.

- First, all those tasks which have been completed and are due for the minute level aggregation are going to be counted.
- Second, depending on the codes or DDIs some of the values will be aggregated by summing and others by counting, maximizing and minimizing.
- Third, the aggregated rows are stored in the record set or CURSOR and then manipulated one by one.
- Fourth, before an aggregated row is inserted in the Datalog table, a new Timeid with a higher level of granularity is required. It will represent the starting time of each task at a higher granularity level than the available one. However, before generating the new Timeid, one of the methods will check whether that Timeid already exists or not. If it exists, then the new Timeid will not be generated, instead the existing one will be retrieved.
- Fifth, the newly created aggregated rows in the DataLog table along with the newly created or retrieved Timeid will be inserted into the DataLog table.
- Last, all the rows whose data is already being aggregated will be deleted from the Datalogvalue table.

```
CREATE PROCEDURE aggregate_data(task int)
MODIFIES SQL DATA
BEGIN
  DECLARE value INT;
  DECLARE codeno INT;
  DECLARE code INT;
  DECLARE total_codes INT;
  SET codeno = 1;
  SET total_codes = count_codes();
     WHILE (codeno <= total_codes) DO
      SET code = get_code_value(codeno);
       CASE
        WHEN code = 0001 OR code = 0002  OR code = 0025
        THEN CALL calculate_max_and_insert(task, code);
        WHEN code = 0050 OR code = 0051 OR code = 0074
        OR code = 0075 OR code = 0077 OR code = 0079
        THEN CALL calculate_sum_and_insert(task, code);
        WHEN code = 0076 OR code = 0078
        THEN CALL calculate_count_and_insert(task, code);
       END CASE;
      SET codeno = codeno + 1;
     END WHILE;
  END

CREATE PROCEDURE calculate_max_and_insert_data(task int,
code int)
MODIFIES SQL DATA
BEGIN
  DECLARE ddi INT;
  DECLARE aggregated_value INT;
  DECLARE time INT;
  DECLARE taskno INT;
  DECLARE time_value INT;
  DECLARE newtimeid INT;
  DECLARE aggregate CURSOR FOR
    SELECT taskid, timeid, datalogDDI, MAX(datalogvalue),
    FROM datalog, timegranularity
    WHERE datalogDDI = code
    AND taskid = task
    AND timeid = timeid
    GROUP BY year, qtr, month, week, day,
             partofday, 4hour, hour, 20minute,
             10minute, 2minute, minute;
  OPEN aggregate;
  REPEAT
     FETCH aggregate INTO taskno, time, ddi,
                          aggregated_value;
     SET time_value =  checktimeid(time);
     IF time_value IS NULL THEN
        CALL inserttime(time_value);
        SET newtimeid = lastvalueoftime();
      ELSE SET newtimeid = time_value;
      END IF;
      INSERT INTO datalog(Taskid, Timeid, DatalogDDI,
                                    Datalogvalue)
      VALUES(taskno, time, ddi, aggregated_value);
   UNTIL DONE END REPEAT;
END
```

The following example will help in understanding how a task could be aggregated. For the time being, we look at task 1 only, in total it ran for 120 seconds. Before calling the aggregation methods, it is of interest to see the data in the Datalog and Timegranularity tables, shown in Table 2 and 3. These are the tables which are going to be affected by the aggregation methods later. In the Timegranularity table (Table 2), the initial granularity is at the second level. It has 120 rows in total; the obvious reason to have 120 rows is due to the nature of task 1 which ran for 120 seconds. So, there is a row to represent each passing second. The rows in the Timegranularity table are already explained in sub-section 3.1.

Table 2. Timegranularity table before aggregation

Tid	Y	Q	M	D	POD	4H	H	20M	10M	2M	M	S	Gran
1	2009	1	4	2	Day[16-00[	[16-20[	19	[20-40[	[20-30[	[28-30[	28	1	S
..	..	.	..	.	..	..	..	..	..	..	..	..	..
60	2009	1	4	2	Day[16-00[	[16-20[	19	[20-40[	[20-30[	[28-30[	28	60	S
61	2009	1	4	2	Day[16-00[	[16-20[	19	[20-40[	[20-30[	[28-30[	29	1	S
..	..	.	..	.	..	..	..	..	..	..	..	..	..
120	2009	1	4	2	Day[16-00[	[16-20[	19	[20-40[	[20-30[	[28-30[	29	60	S

Table 3. Datalog table before aggregation

Taskid	Timeid	DatalogDDI	Datalogvalue
1	1	0002	268
1	1	0001	1000
1	1	0074	0
..	..	..	..
1	120	0075	54
1	120	0076	18
1	120	0078	28
1	120	0077	31

Whereas, in the Datalog table there are 11 different DDIs with their values and references to Timeid in the Timegranularity table as well as to Taskid in the Task table. For example, row number 1 in the Datalog table (Table 3) reads as follows: Taskid = 1 (represents: id for a task), Timeid = 1 (represents: 19:28:01 02.04.2009), DatalogDDI = 0002 (represents: setpoint volume per area application rate) and Datalogvalue = 268 (represents: actual value).

After the minute level aggregation method is being called, it will aggregate the data which is older than three months; in the given example whole data set is older than three months, so the following changes in the Timegranularity and Datalog tables have taken place. New rows have been added with the minute level time granularity in the Timegranularity table, as shown in the Table 4. Further, new rows have also been added in the Datalog table, as shown in the Table 5, these new rows contain aggregated data (per task per minute level). Depending on the DDIs, values will be aggregated as SUM, MAX and COUNT. Furthermore, all the rows containing detailed data, which is aggregated, have been deleted.

As seen, in the Timegranularity table (Table 4) after minute level aggregation, two new rows have been added, rows with Timeid 121 and 122. As the granularity of these newly added rows is at minute level, for that reason the values after Minute attribute are NULL. Also, Timeid 121 and 122 have been used in the Datalog table (Table 5) to represent the newly aggregated values at the minute granularity level.

Furthermore, after data is being aggregated at the minute level, we could also call the 2 minutes level aggregation method if the data fulfill the time requirements, such as the data should be older than six months that is also the case in this example. After calling the 2 minutes level aggregation method the following changes in the Timegranularity and Datalog tables have taken place. New rows have been added with the 2 minutes level time granularity in the Timegranularity table, as shown in the Table 6.

Moreover, new rows have also been added in the Datalog table (Table 7), these new rows contain aggregated data (at per task per 2 minutes level). Depending on the DDIs some of those values will be aggregated as maximums, other as counts and sums. Furthermore, all the rows containing detailed data, which is aggregated, have been deleted. Table 7 shows the final result of the aggregation methods.

To summarize the aggregation process, we started with Table 3, which has 1320 rows in total, then after first phase of data aggregation from the second granularity level to the minute granularity level, Table 5 is generated with 22 rows, 98.34% reduction (from the starting rows 1320). After second phase of data aggregation from the minute granularity level to the 2 minutes granularity level, Table 7 is generated with only 11 rows, 99.17% reduction (from the starting rows 1320). The data aggregation process is responsible not only for huge reduction in the volume of data over time but also the aggregated data can be used for analysis and planning purposes.

Table 4. Timegranularity table after minute level aggregation

Tid	Y	Q	M	D	POD	4H	H	20M	10M	2M	M	S	Gran
..	..	..	..	..	..	..	..	..	..	..	..	..	..
120	2009	1	4	2	Day[16-00[	[16-20[	19	[20-40[	[20-30[	[28-30[	18	60	S
121	2009	1	4	2	Day[16-00[	[16-20[	19	[20-40[	[20-30[	[28-30[	28	NULL	M
122	2009	1	4	2	Day[16-00[	[16-20[	19	[20-40[	[20-30[	[28-30[	29	NULL	M

Table 5. Datalog table after minute level aggregation

Taskid	Timeid	DatalogDDI	Datalogvalue
1	121	0002	298
1	121	0001	1000
1	121	0074	1017
..	..	..	..
1	122	0075	54
1	122	0076	76
1	122	0078	57
1	122	0077	31

Table 6. Timegranularity table after 2 minutes level aggregation

Tid	Y	Q	M	D	POD	4H	H	20M	10M	2M	M	S	Gran
..	..	.	..	..	..	..	..	..	..	..	..	..	..
120	2009	1	4	2	Day[16-00[	[16-20[	19	[20-40[	[20-30[	[28-30[	28	60	S
121	2009	1	4	2	Day[16-00[	[16-20[	19	[20-40[	[20-30[	[28-30[	28	NULL	M
122	2009	1	4	2	Day[16-00[	[16-20[	19	[20-40[	[20-30[	[28-30[	29	NULL	M
123	2009	1	4	2	Day[16-00[	[16-20[	19	[20-40[	[20-30[	[28-30[	NULL	NULL	2M

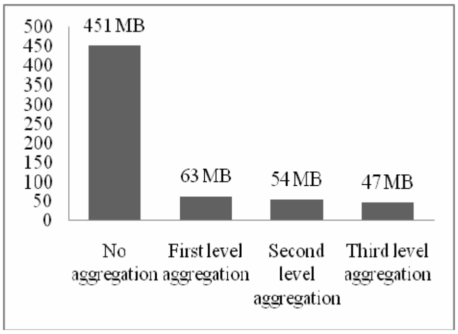
Table 7. Datalog table after 2 minutes level aggregation

Taskid	Timeid	DatalogDDI	Datalogvalue
1	123	0002	328
1	123	0001	900
1	123	0074	1721
1	123	0077	31
1	123	0079	1000
1	123	0050	19
..	..	..	..

5 Evaluation

An evaluation of the gradual granular data aggregation solution has been done with the proposed time granularity table. Performance tests have been carried out on single-level aggregation queries. The queries aggregate data gradually from a single lower level of granularity to a higher level of granularity. For example, to aggregate values from *per parameter per task per second* to *per parameter per task per minute*, further to aggregate values from *per parameter per task per minute* to *per parameter per task per 2 minutes* and so on. The tests were designed to measure the initial storage used in MB (before the gradual aggregation process is performed), aggregation speed in seconds, deletion speed in seconds (for rows that have just been aggregated), size of each level of granularity in MB (data reduction after each level of aggregation), overall aggregation speed in seconds and total storage used in MB (after the gradual aggregation process has been performed). The tests were performed on a 2.0 GHz Intel® Core Duo with 512 MB RAM, running Ubuntu 8.04 (hardy) and MySQL 5.0.5. Every test was performed 5 times. The maximum and minimum values are discarded and an average is calculated using the middle three values.

From the tests, it is observed that the initial storage used by the proposed solution (before the gradual aggregation process) to store 10,000,000 rows is approximately 451 MB (Fig. 4a). As a matter of fact, the Datalog table is more critical than the Timegranularity table, since in the Timegranularity table, rows are likely to comprise less than 3 % of the total size of the database, whereas, in the case of Datalog table it is 97 %. The reason for the difference is that the values were logged for 20 DDIs at a granularity of one second per 20 values with approximately 222,222 readings for each DDI and 10 DDIs at a granularity of one minute per 10 values with approximately 111,111 reading for each DDI, which requires approximately 333,333 rows in the Timegranularity table to represents 10,000,000 rows in the Datalog table. The total data consists of 10,000,000 rows (data log values) distributed uniformly across a two-year time period. Out of 10,000,000 rows 1,000,000 rows were less than three months



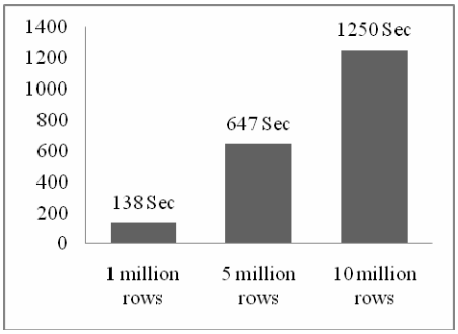


Fig. 4a. Size of each level of granularity

Fig. 4b. Aggregation speed

old; as a result actually 9,000,000 rows were aggregated. The aggregation speed of the single-granular query is 110,000 rows per second and the deletion speed is 45,000 rows per second. The size of the data after the first level of aggregation (from the second to the minute granularity level) is 63 MB. The reduction in storage space at this level is thus 86 %. Similarly, after the second level of aggregation (from the minute to the 2 minutes granularity level) the size of data is 54 MB that indicates a further reduction of 15 %. Finally, after third level of aggregation (from the 2 minutes level to the 10 minutes level) the size of data is 47 MB, which represents an additional reduction of 13 % (Fig. 4a). The reduction in the number of rows at level two and three is fewer as compared to the first level aggregation. The reason is that, the first level aggregation was applied to a larger set of fine-grained data, whereas the second and third level aggregation is applied to a smaller set of coarse-grained data. The complete aggregation process consists of 1) finding the tasks that are due for aggregation, 2) aggregating the existing rows based on single granularity, 3) generating the new rows in the Timegranularity table in order to point to the higher granularity rows in the Datalog table, 4) inserting the newly aggregated rows in the Datalog table and 5) deleting the previous rows from the Datalog table. It takes approximately 1250 seconds to aggregate 6,000,000 rows from the second granularity level to the minute granularity level, insert approximately 100, 000 new aggregated rows and delete approximately 5,900,000 previous rows (Fig. 4b). Finally, the total storage used by the proposed solution (after gradual aggregation process has been per formed) is approximately 47 MB (Fig. 4a) that suggests a total reduction in the storage is 89.5 %. In conclusion, the above mentioned test results have shown that the proposed solution scales reasonably. The reduction in the number of rows is quite significant. The major aspect of the solution is that it maintains the data at different levels of granularity depending on the age of the data. The data at the each level of granularity can be used for analysis and reporting purpose. To the best of our knowledge this is the first paper to conduct a performance evaluation of the gradual data aggregation methods.

6 Related Work

Previously, other studies on data aggregation have been done. A comprehensive survey of most relevant techniques for the evaluation of aggregate queries on spatiotemporal data is presented by [3]. Efficient aggregation algorithms for compressed data warehouses are proposed by [4]. Techniques such as pattern identification, categorization, feature extraction, drift calculation and generalization for the aggregation of information are summarized in [5]. Multi-dimensional extension of the ER model to use aggregated data in complex analysis contexts is proposed by [6]. However, the main objective of these approaches is to perform one time aggregation of data rather than gradual aggregation, as presented in this paper. In the context of gradual data aggregation work has been reported. An efficient tree based indexing schemes for gradually maintaining aggregates is presented in [7]. The focal point of this work is on presenting effective indexing schemes for storing aggregated data. A language for specifying a strategy to archive data and keep summaries of archived data in data warehouses is presented by [8]. Further, the semantic foundation for data reduction in data warehouses that permits the gradual aggregation of detailed data as the data gets older is provided by [1]; however, both of these related works are highly theoretical. Finally, a concept for gradual data aggregation in multi-granular databases has been described in [9]; though, a concrete example and an implementing strategy are missing. In comparison to most of the above mentioned approaches, our work concentrates on all aspects of gradual data aggregation solution that includes conception, implementation and evaluation.

7 Conclusion

This paper proposed an effective solution for data reduction. The data reduction technique is based on a gradual granular data aggregation mechanism. By using the proposed solution, not only data is kept for long time periods by keeping the newer data as fine-grained and the older data as coarse-grained, but also major gains of 70 -90 % reduction in storage space are achieved. To save storage space is important not only for the farming industry but for any other type of industry in which significant amounts of data are generated. In future research, the direction of interest is to implement a high level rule-based tool for gradual granular data aggregation. It is also interesting to investigate, how to remove the redundancies from the Timegranularity table.

Acknowledgements. This work is supported by the LandIT project, and is funded by the Danish Ministry of Science, Technology and Innovation.

References

1. Skyt, J., Jensen, C.S., Pedersen, T.B.: Specification-based Data Reduction in Dimensional Data Warehouses. Information Systems 33(1), 36–63 (2008)
2. LandIT, `http://www.tekkva.dk/page326.aspx`
3. Lopez, I.F.V., Moon, B., Snodgrass, R.T.: Spatiotemporal Aggregate Computation: A Survey. IEEE Transactions on Knowledge and Data Engineering 17(2), 271–286 (2005)
4. Li, J., Srivastava, J.: Efficient Aggregation Algorithms for Compressed Data Warehouses. IEEE Transactions on Knowledge and Data Engineering 14(3), 515–529 (2002)

5. Rasheed, F., Lee, Y.K., Lee, S.: Towards using Data Aggregation Techniques in Ubiquitous Computing Environments. In: 4th IEEE International Conference on Pervasive Computing and Communication Workshops, pp. 369–392. IEEE Press, New York (2006)
6. Schulze, C., Spilke, J., Lehner, W.: Data Modeling for Precision Dairy Farming within the Competitive Field of Operational and Analytical Tasks. Computers and Electronics in Agriculture 59(1-2), 39–55 (2007)
7. Zhang, D., Gunopulos, D., Tsotras, V.J., Seeger, B.: Temporal and Spatio-Temporal Aggregations over Data Streams using Multiple Time Granularities. Information Systems 28(1-2), 61–84 (2003)
8. Boly, A., Hébrail, G., Goutier, S.: Forgetting Data Intelligently in Data Warehouses. In: IEEE International Conference on Research, Innovation and Vision for the Future, pp. 220–227. IEEE Press, New York (2007)
9. Iftikhar, N.: Integration, Aggregation and Exchange of Farming Device Data: A High Level Perspective. In: 2nd IEEE Conference on the Applications of Digital Information and Web Technologies, pp. 14–19. IEEE Press, New York (2009)

Automation of the Medical Diagnosis Process Using Semantic Image Interpretation

Anca Loredana Ion and Stefan Udristoiu

Faculty of Automation, Computers, and Electronics, Bvd. Decebal, Nr. 107, 200440, Craiova, Romania
{anca_soimu,s_udristoiu}@yahoo.com
http://www.software.ucv.ro

Abstract. This paper is a part of a complex study of developing methods for semantic interpretation of medical images, to permit the semi-automatic diagnosis. The first objective of the study is to develop new methods for medical image segmentation and a set of visual features. The second objective consists of developing a unifying framework for semantic images annotation, to be used in the process of medical diagnosis. The developed diagnosis method is based on on semantic pattern rules capable to discover associations between visual features of medical images and their diagnoses. Although we present the results achieved in endoscopic images analysis, our methods can be used to analyze other types of medical images. The prototype system was applied to real datasets and the results show high accuracy.

Keywords: medical image diagnosis, image colour, image texture, image shape, semantic association rules, image mining.

1 Introduction

In recent years, significant advances have been made in the area of automatic extraction of low-level features from visual content of medical images. However, little progress has been achieved in the identification of high-level semantic features or the effective combination of semantic features derived from different modalities [1], [11].

An impressive amount of medical images is daily generated in hospitals and medical centers. Consequently, the physicians have an increasing number of images to analyze manually. Computational techniques can be provided to assist the physician's work, as computer assisted diagnosis systems, which support physicians in analyzing digital images and to find out possible diseases [3], [4].

For medical applications of CBIR, an earlier overview was given by Müller [2] in 2004. Regarding image analysis, there are numerous advanced object recognition algorithms for the detection of specific organs and structures from medical images. However, existing image analysis methodologies are not generic enough to directly handle different organs or imaging modalities outside of the original application domain.

In the recent past, applications like FIRE [11] showed that image retrieval based only on sub-symbolic interpretation of the images works well for a number of applications.

In the IRMA project [12], this technology was applied to the medical domain. A number of researches use ontology-based image retrieval to emphasize the necessity to combine sub-symbolic object recognition and abstract domain knowledge.

In [13] an integration of spatial context and semantic concepts into the feature extraction and retrieval process using a relevance feedback procedure is proposed.

In [14] a system that aims at applying a knowledge-based approach to interpret X-ray images of bones and to identify the fractured regions is proposed. In [15] a hybrid method which combines symbolic and sub-symbolic techniques for the annotation of brain Magnetic Resonance images is developed. In [6], [7], relevant methods using association rules for radiology image analysis are developed.

In this paper, we propose a method that employs semantic pattern rules to support computer assisted diagnosis systems. The rules reflect how humans analyze and combine the visual features to determine the images diagnosis. In this paper, we will show that pattern rules can be successfully applied to support medical image diagnosis.

The remainder of this paper is structured as follows. Section 2 presents the selection of visual features; section 3 presents the mapping between visual features and diagnosis; section 4 presents the generation of semantic rules and details the process of medical diagnosis based on semantic pattern rules. Finally, section 5 discusses the experiments and Section 6 summarizes the conclusions of this study.

2 Visual Representation of Images

Many techniques have been proposed and many prototype systems have been developed to solve the problems in retrieving images according to their visual image content. One problem of the feature-based technique is that there is a considerable 'gap' between users' interest in reality and the image contents described by the low-level perceptive features.

The selection of the visual feature set and the image segmentation algorithm are the definitive stage for establishing the diagnosis of medical colour images. The diagnosis of medical images is directly related to the visual features (colour, texture, shape, position, dimension, etc.) because these attributes capture the information about the semantic meaning.

A set of colour regions is obtained from each image by segmentation after colour characteristic [9]. The specialist selects the minimum rectangle (MR), which bounds the sick regions of an image. In the area bounded by the minimum selected rectangle, the colour-set back projection algorithm [8] detects the

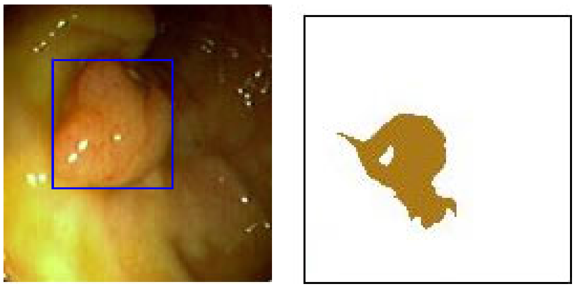


Fig. 1. Segmentation results from the image diagnosed with colon cancer

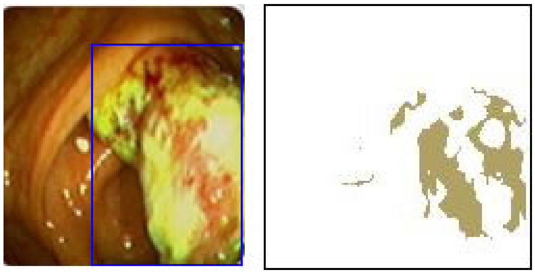


Fig. 2. Segmentation results from an image diagnosed with colon cancer

adjacent pixels of the same colour. The results of the segmentation algorithm applied to two images diagnosed with colon cancer [17] can be visualized in Figure 1 and Figure 2.

In conformity with the defined characteristics [9], a region is described by:

- The colour, which is represented in the HSV colour space quantized at 166 colours. A region is represented by a colour index which is, in fact, an integer number between 0..165.
- The spatial coherency, which measures the spatial compactness of the pixels of the same colour.
- A seven-dimension vector (maximum probability, energy, entropy, contrast, cluster shade, cluster prominence, correlation), which represents the texture characteristics.
- The region dimension descriptor, which represents the number of pixels from region.
- The spatial information which is represented by the centroid coordinates of the region and by minimum bounded rectangle.

– A two-dimensional vector (eccentricity and compactness), which represents the shape feature.

3 Visual Semantic Indicators

The developed vocabulary is based on the concept of semantic indicators, and the syntax captures the basic models about patterns and diagnosis. The proposed representation language is simple, because the syntax and vocabulary are elementary.

The vocabulary words are limited to the name of semantic indicators. Being visual elements, the semantic indicators are, by example, the colour (colour-light-red), spatial coherency (spatial coherency-weak, spatial coherency-medium, spatial coherency-strong), texture (energy-small, energy-medium, energy-big, etc.), dimension (dimension-small, dimension-medium, dimension-big, etc.), position (vertical-upper, vertical-center, vertical-bottom, horizontal-upper, etc.), shape (eccentricity- small, compactness-small, etc.) [16].

The syntax is represented by the model, which describes the images in terms of semantic indicators values. The values of each semantic descriptor are mapped to a value domain, which corresponds to the mathematical descriptor. The value domains of visual characteristics were manually experimented on images of WxH dimension. A value of colour semantic indicator is associated to each region colour in the HSV colour space quantized at 166 colours. The colour correspondence between the mathematical and semantic indicator values is determined based on the experiments effectuated on a training image database. The colour correspondence is illustrated by the following examples: light-red (108), medium-red (122), dark-red (139), light-yellow (109), medium-yellow (125), dark-yellow (141).

Similarly, a hierarchy of values, which are mapped to semantic indicator values, is also determined for the other visual characteristics.

At the end of the mapping process, a medical image is represented in Prolog by means of the terms *figure(ListofRegions)*, where *ListofRegions* is a list of image regions [16]. The term *region(ListofDescriptors)* is used for region representation, where the argument is a list of terms used to specify the semantic indicators. The term used to specify the semantic indicators is of the form:

$$descriptor(DescriptorName,\ DescriptorValue).$$

The mapping between the values of low-level (mathematical) descriptors and the values of semantic indicators is based on experiments effectuated on images from different diagnoses and the following facts are used:

$$mappingDescriptor(Name, SemanticValue, ListValues).$$

The argument *Name* is the semantic indicator name, *SemanticValue* is the value of the semantic indicator, *ListValue* represents a list of mathematical values and closed intervals, described by the following terms: *interval(InferiorLimit, SuperiLimit).*

4 Determining Image Medical Diagnosis Using Semantic Pattern Rules

A semantic pattern rule is of form:

$$\textit{semantic indicators} \rightarrow \textit{diagnosis}$$

The semantic pattern rules have the body composed by conjunctions of semantic indicators, while the head is the diagnosis. The steps of the learning process are:

– relevant images are used to learn a diagnosis,
– each image is automatically segmented and the primitive visual features are computed,
– for each image, the primitive visual features are mapped to semantic indicators,
– the rule generation algorithm is applied to produce rules, which will identify each diagnosis from the database.

The image testing phase has as scope automatic diagnosing of images:

– each new image is processed and segmented into colour regions,
– for each new image the low-level characteristics are mapped to semantic indicators,
– the classification algorithm is applied for identifying the image diagnosis.

4.1 Semantic Pattern Rule Generation

In our system, the learning of semantic pattern rules is continuously made, because when a diagnosed image is added in the learning database, the system continues the process of rules generation.

The image pattern rules have to find the semantic relationships between image objects and diagnosis. The proposed method uses the Apriori algorithm for discovering the semantic pattern rules between primitive characteristics extracted from images and diagnosis, which images belong to [7]. Before producing semantic rules based on region patterns, a pre-processing phase for determining the visual similitude between the image regions from the same diagnosis is necessary.

In the preprocessing phase, the region patterns, which appear in the images, are determined. So, each j region of an i image, Reg_{ij}, is compared with other image regions with the same diagnosis. If the region Reg_{ij} matches the m region of the k image, Reg_{km}, having similar features on the positions n_1, n_2,..,n_c, then the generated region pattern is RS(-, -, -, n_1, n_2,..,n_c,-,-), and the other features are ignored. The algorithm that generates image region patterns is described in pseudo-code:

```
Algorithm. Generation of region patterns for each image.
Input: set of images {I1, I2,..,In} laying in the same diagnosis;
       each image Ii is a set of regions Regim, where  i = 1..n,
       and n is the number of images with the same diagnosis,
       m=1..Ii.nregions, and Ii.nregions is the regions number of
       the Ii image.
Output: set of region patterns RSi for each image Ii.
Method:
for i1 = 1, n-1 do {
    for j1 = 1, Ii1.nregions do {
        for i2 = i1 + 1, n do {
            for j2 = 1, Ii2.nregions do{
                if Regi1j1  matches Regi2j2  in the
                    components n1,n2,.., nc  then {
                    RSi1 = RSi1 + RS(-,-,n1,n2,..,nc,-,-)
                    RSi2 = RSi2 + RS(-,-,n1,n2,..,nc,-,-)
                }
            }
        }
    }
}
```

A database with five images, relevant for a diagnosis, is considered. An example of image representations using region patterns is presented in Table 1.

Table 1. Relevant images for a certain diagnosis

ID	Image Regions
1	R(a,b,c,d), R(a',b',c',d')
2	R(a,b,c,f), R(a',b', c',d')
3	R(a,b,c,f"), R(a',b',c',i)
4	R(a,b,c,f'), R(a',b',c',i')
5	R(a,b,e,d), R(m,b',c',d')

Table 2. Region patterns of images from a certain diagnosis

ID	Region Patterns
1	RS(a,b,c,-), RS(a,b,-,d), RS(a',b',c',d'), RS(a',b',c',-), RS(-,b',c',d')
2	RS(a,b,c,-), RS(a,b,-,-), RS(a',b',c',d'), RS(a',b',c',-), RS(-,b',c',d')
3	RS(a,b,c,-), RS(a,b,-,-), RS(a',b',c',-), RS(-,b',c',-)
4	RS(a,b,c,-), RS(a,b,-,-), RS(a',b',c',-), RS(-,b',c',-)
5	RS(a,b,-,d), RS(a,b,-,-), RS(-,b',c',d'), RS(-,b',c',-).

By determining the region patterns, the results from the Table 2 are obtained.
The image modeling in terms of itemsets and transactions is the following:

- the transactions represent the set of images represented as region patterns, determined by the previous algorithm.
- the itemsets are formed by region patterns of the images laying in the same diagnosis.
- the frequent itemsets represent the itemsets with the support bigger than the minimum support.
- the itemsets of cardinality between 1 and k are iteratively found, where k represents the maximum length an itemset.
- the frequent itemsets are used for rule generation, with $k > 1$.
- each semantic pattern rule is associated to support that represents the percent of transactions, in which both the body and head of the rule appear, and to confidence that represents the ratio between the number of transactions, in which both the body and the head appear, and the number of transactions, in which only the body appears.

The algorithm for rules generation based on region patterns is described in pseudo-code:

```
Algorithm. Rule generation on the training set of the transactional
           database with the collection divided in subsets
           by diagnosis.
Input: a set of |D| dimension including images with a diagnosis di;
       each image is represented as a set of region patterns.
Output: the set of semantic pattern rules for diagnosis di.
Method:
       Ck: itemsets of k-length.
       Lk: itemsets of k-length.
       Rulesi: the set of rules constructed from
               the frequent itemsets for k>1.
       L1= {frequent region patterns};
        for(k = 1; Lk != null; k++)
        {
            Ck+1=itemsets generated from the set Lk.
            foreach transaction t in the database
            {
                /* Increment by 1 the number of all itemsets
                   that appear in t.*/
                count(Ck+1) = count(Ck+1)+1
                /* Compute the support of itemsets */
                sup(Ck+1) =  (count(Ck+1)/|D|)*100
              }
                Lk+1 = candidates from Ck+1 that has the
                     support greater than suport_min.
                Rulesi = Rulesi + {Lk+1->di}
            }
```

Table 3. Step 1 - Generation of itemsets of cardinality one

Itemsets	Support(%)
RS(a,b,c,-)	80
RS(a,b,-,d)	40
RS(a',b',c',d')	40
RS(a',b',c',-)	80
RS(-,b',c',d')	60
RS(a,b,-,-)	80
RS(-,b',c',-)	60

Table 4. The frequent itemsets of cardinality one

Itemsets	Support(%)
RS(a,b,c,-)	80
RS(a',b',c',-)	80
RS(-,b',c',d')	60
RS(a,b,-,-)	80
RS(-,b',c',-)	60

Table 5. Step 2 - Generation of itemsets of cardinality two

Itemsets	Support(%)
RS(a,b,c,-), RS(a',b',c',-)	80
RS(a,b,c,-), RS(-,b',c',d')	40
RS(a,b,c,-), RS(a,b,-,-)	60
RS(a,b,c,-), RS(-,b',c',-)	40
RS(a',b',c',-), RS(-,b',c',d')	40
RS(a',b',c',-), RS(a,b,-,-)	60
RS(a',b',c',-), RS(-,b',c',-)	40
RS(-,b',c',d'), RS(a,b,-,-)	40
RS(-,b',c',d'), RS(-,b',c',-)	20
RS(a,b,-,-), RS(-,b',c',-)	60

Applying the algorithm on transaction set from Table 2, the following results are obtained (the minimum support is 40%, and the minimum confidence is 70%). Table 3 contains the itemsets of one-length.

Table 4 contains the frequent itemsets of cardinality one, that have the support bigger than the minimum support.

Table 5 contains the itemsets of two-length generated from the itemsets from Table 4.

Table 6 contains the frequent itemsets of cardinality two, that have the support bigger than the minimum support.

Table 7 contains the itemsets of three-length generated from the itemsets from Table 6.

Table 6. The frequent itemsets of cardinality two

Itemsets	Support(%)
RS(a,b,c,-), RS(a',b',c',-)	80
RS(a,b,c,-), RS(a,b,-,-)	60
RS(a',b',c',-), RS(a,b,-,-)	60
RS(a,b,-,-), RS(-,b',c',-)	60

Table 7. Step 3 - Generation of itemsets of cardinality three

Itemsets	Support(%)
RS(a,b,c,-), RS(a',b',c',-), RS(a,b,-,-)	60
RS(a,b,c,-), RS(a',b',c',-), RS(-,b',c',-)	40
RS(a',b',c',-), RS(a,b,-,-), RS(-,b',c',-)	40

Table 8. The frequent itemsets of cardinality three

Itemsets	Support(%)
RS(a,b,c,-), RS(a',b',c',-), RS(a,b,-,-)	60

The patterns set from Table 8 contains the frequent itemsets of cardinality three, that have the support bigger than the minimum support.

The diagnosis d_i of images from Table 1 is determined by the following semantic pattern rules. A semantic pattern rule is composed by conjunctions of region patterns of k cardinality ($k > 1$):

RS(a,b,c,-) and RS(a',b',c',-) $\rightarrow d_i$,
RS(a,b,c,-) and RS(a,b,-,-) $\rightarrow d_i$,
RS(a',b',c',-) and RS(a,b,-,-) $\rightarrow d_i$,
RS(a,b,-,-) and RS(-,b',c',-) $\rightarrow d_i$,
RS(a,b,c,-) and RS(a',b',c',-) and RS(a,b,-,-) $\rightarrow d_i$.

At the end of the generation of rule sets for each diagnosis d_i from the database, the rule confidence is computed:

```
Algorithm. Compute the rule confidence.
Input: Sets of Rulesi, where each rule has the representation:
       body(rule) determines head(rule).
Output: c, confidence of each rule.
Method:
        Rules = union all the rules {Rulesi}
        foreach rulek from Rules
        {
            N1k=number of rules that contain body(rulek) and
                head(rulek).
            N2k=number of rules that contain body(rulek).
            conf(rulek) = N1k/N2k
        }
```

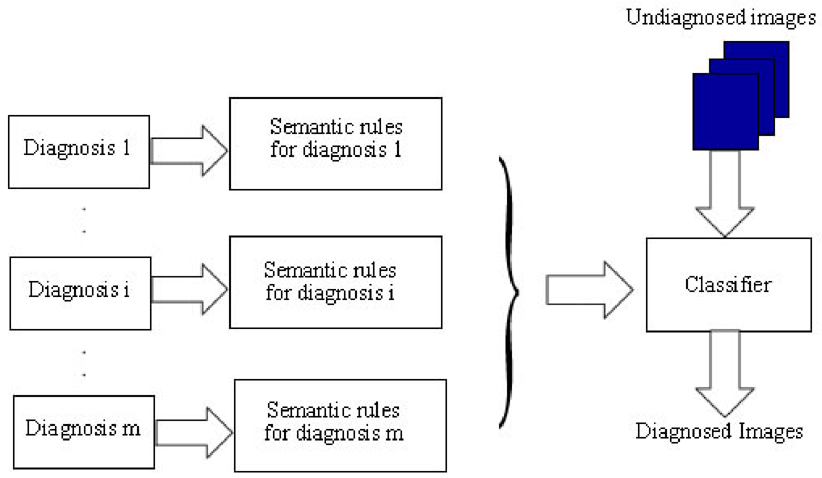

Undiagnosed images
Diagnosis 1
Semantic rules for diagnosis 1
Diagnosis i
Semantic rules for diagnosis i
Diagnosis m
Semantic rules for diagnosis m
Classifier
Diagnosed Images

Fig. 3. Image semantic classifier

4.2 Medical Image Annotation

The set of generated rules $Rules_i$ (the i^{th}- diagnosis from the database) represent the classifier. The classifier is used to predict which diagnosis the images from test database belong to. Being given a new image, the classification process searches in the rules set for finding its most appropriate diagnosis, as in Figure 3.

A semantic pattern rule matches an image if all characteristics, which appear in the body of the rule, also appear in the image characteristics. The semantic pattern rules with maximum confidence are selected.

```
Algorithm. Semantic classification of an image.
Input: new unclassified image and the set of semantic pattern rules;
       each pattern rule has the confidence Ri.conf.
Output: the classified image, and the score of matching.
Method:
        S = null
        foreach rule R in Rules do{
        begin
            if R matches I then {
                *Keep R and add R in S
                I.score = R.conf
                *Divide S into subsets one
                foreach category: S1,..,Sn do{
                    foreach subset Sk from S do {
                       *Add the confidences of all rules from Sk.
                        *Add I image in the category identified by
                        the rules from Sk with the greatest confidence.
                        I.score = maxSk.conf
                    }
                }
             }
        }
```

5 Experiments

In the experiments realized through this study, two databases are used for learning and testing. The database used to learn the correlations between images and diagnoses, contains 400 images with digestive diagnoses.

The learning database is categorized into the following diagnoses: colon cancer, ulcer, polyps, gastric cancer, esophagitis, and rectocolitis. The system learns each concept by submitting about 20 images per diagnosis. For each diagnosis, the following metrics (accuracy-A, sensitivity-S, specificity-SP) are computed:

$$A = \frac{TP + TN}{TP + FP + FN + TN} \ . \tag{1}$$

$$S = \frac{TP}{TP + FN} \ . \tag{2}$$

$$SP = \frac{TN}{FP + TN} \ . \tag{3}$$

where, TP represents the number of true positives (images correctly diagnosed with the searched diagnosis), FP represents the number of false positives (images incorrectly diagnosed with the searched diagnosis), TN represents the number of true negatives (images correctly diagnosed with a different diagnosis), FN represents the number of false negatives (images incorrectly diagnosed with a different diagnosis.

The results of the presented method are very promising, being influenced by the complexity of endoscopic images as can be observed in Table 9. Improvements can be brought using a segmentation method with greater semantic accuracy.

Table 9. Recorded results

Diagnosis	Accuracy(%)	Sensitivity(%)	Specificity(%)
Ulcer	96.5	92.5	71
Polyps	96.5	92	71.5
Esophagitis	96.0	90.5	71
Gastric Cancer	96.1	90.5	71.5
Rectocolitis	97.1	93.7	72.9
Colon Cancer	97.1	92.3	72.5

6 Conclusion

Methods proposed and developed in this study could assist physicians by doing automatic diagnosis based on visual content of medical images. An important feature of these methods is their general applicability, because the semantic concepts and the rules from images can be learned in any domain.

In the future work, we try to resolve the problems of the current methods for interpreting the image diagnoses: the improvement of the semantic classifiers accuracy, the semantic concept description and understandability such that the relations between the low-level visual features and the semantic concepts of the diagnosis could be more clear. We have to improve the methods of selecting the visual feature value sets that can well capture the semantic concepts of medical images.

Actually, the results of experiments are very promising, because they show a good accuracy and sensitivity and a medium precision for the majority of the database categories, making the system more reliable.

Acknowledgment

This work was supported by the strategic grant POSDRU/89/1.5/S/61968, Project ID 61968 (2009), co-financed by the European Social Fund within the Sectorial Operational Program Human Resources Development 2007-2013,title of the project "Methods for automatic diagnosis of medical images, assisted by computer".

References

1. Zhou, X.S., Zillner, S., Moeller, M., Sintek, M., Zhan, Y., Krishnan, A., Gupta, A.: Semantics and CBIR: a medical imaging perspective. In: Proceedings of the 2008 International Conference on Content-Based Image and Video Retrieval (2008)
2. Müller, H., Michoux, N., Bandon, D., Geissbuhler, A.: A review of content-based image retrieval systems in medical applications clinical benefits and future directions. International Journal of Medical Informatics 73(1), 1–23 (2004)
3. Smeulders, A.W.M., Worring, M., Santini, S., Gupta, A., Jain, R.: Content-based image retrieval at the end of the early years. IEEE Trans. Pattern Anal. Machine Intelligence. 22(12), 1349–1380 (2000)
4. Duncan, J., Ayache, N.: Medical image analysis: Progress over two decades and the challenges ahead. IEEE Trans. Pattern Anal. Machine Intelligence 22(1), 85–106 (2000)
5. Ribeiro, M.X., Traina, A.J., Rosa, N.A., Marques, P.M.: How to Improve Medical Image Diagnosis through Association Rules: The IDEA Method. In: Proceedings of the 2008 21st IEEE International Symposium on Computer-Based Medical Systems (2008)
6. Antonie, M.L., Zaane, O.R., Coman, A.: Associative classifiers for medical images. In: Zaïane, O.R., Simoff, S.J., Djeraba, C. (eds.) MDM/KDD 2002 and KDMCD 2002. LNCS (LNAI), vol. 2797, pp. 68–83. Springer, Heidelberg (2003)
7. Pan, H., Li, J., Wei, Z.: Mining interesting association rules in medical images. Advance Data Mining and Medical Applications (2005)
8. Smith, J.R., Chang, S.-F.: VisualSEEk: a fully automated content-based image query system. In: The Fourth ACM International Multimedia Conference and Exhibition, Boston, MA, USA (1996)
9. Ion, A.L.: Methods for Knowledge Discovery in Images. Information Technology and Control 38(1), 43–49 (2009)
10. Liu, Y., Zhang, D., Lu, G., Ma, W.-Y.: A survey of content-based image retrieval with high-level semantics. Pattern Recognition 40(1), 262–282 (2007)

11. Deselaers, T., Keysers, D., Ney, H.: FIRE-flexible image retrieval engine: ImageCLEF 2004 evaluation. In: Peters, C., Clough, P., Gonzalo, J., Jones, G.J.F., Kluck, M., Magnini, B. (eds.) CLEF 2004. LNCS, vol. 3491, pp. 688–698. Springer, Heidelberg (2005)
12. Lehmann, T., Güld, M., Thies, C., Fischer, B., Spitzer, K., Keysers, D., Ney, H., Kohnen, M., Schubert, H., Wein, B.: The IRMA project. A state of the art report on content-based image retrieval in medical applications. In: Proceedings 7th Korea-Germany Joint Workshop on Advanced Medical Image Processing, pp. 161–171 (2003)
13. Vompras, J.: Towards adaptive ontology-based image retrieval. In: Stefan Brass, C.G. (ed.) 17th GI-Workshop on the Foundations of Databases, Wrlitz, Germany, pp. 14–152 (2005)
14. Su, L., Sharp, B., Chibelushi, C.: Knowledge- based image understanding: A rule-based production system for X-ray segmentation. In: Proceedings of Fourth International Conference on Enterprise Information System, Ciudad Real, Spain, pp. 530–533 (2002)
15. Mechouche, A., Golbreich, C., Gibaud, B.: Towards an hybrid system using an ontology enriched by rules for the semantic annotation of brain MRI images. In: Marchiori, M., Pan, J.Z., Marie, C.d.S. (eds.) RR 2007. LNCS, vol. 4524, pp. 219–228. Springer, Heidelberg (2007)
16. Ion, A.L.: Image Annotation Based on Semantic Rules. In: Human-Computer Systems Interaction: Backgrounds and Applications, pp. 83–97. Springer, Heidelberg (2009)
17. Jackson Siegelbaum Gastroenterology, http://gicare.com/Endoscopy-Center/Endoscopy-images.aspx

A Language for Ontology-Based Metamodeling Systems

Stéphane Jean, Yamine Aït-Ameur, and Guy Pierra

LISI-ENSMA and University of Poitiers
BP 40109, 86961 Futuroscope Cedex, France
{jean,yamine,pierra}@ensma.fr

Abstract. Nowadays, metamodeling techniques and ontologies play a central role in many computer science problems such as data exchange, integration of heterogeneous data and models or software reuse. Yet, if many metamodeling repositories and ontology repositories have been proposed, few attempts have been made to combine their advantages into a single repository with a dedicated language. In this paper, we present the capabilities of the OntoQL language for managing the various levels of information stored in a metamodeling systems where (1) both data, models and metamodels are available and (2) semantic descriptions are provided using ontologies. Some of the main characteristics of OntoQL are: management of the metamodel level using the same operators as the ones applied to data and to model levels, semantic queries through ontologies and SQL compatibility. We report several applications where this language has been extensively and successfully used.

1 Introduction

In the last two decades, with the appearance of model driven engineering techniques where both data, models and mappings between models are represented as data, metadata play a major role and become central concepts manipulated by these techniques. They are used in the definition of computer based solutions for interoperability, data exchange, software reuse, model transformation and so on. At the origin, databases with their system catalogs and functional languages with continuations (like in Lisp) were the first systems that permitted the explicit manipulation of these metadata.

When designing information systems, the introduction of metadata leads to information systems whose architecture is composed of three levels representing: data, models and metamodels. Such systems are named metamodel based systems (MMS). Three domains present relevant examples putting into practice MMS systems: 1) ontologies, 2) database and 3) software engineering.

1. In the semantic web, ontology models are used for representing ontologies. Yet, these ontologies are themselves models of their instances. As a consequence, managing these data actually requires the three levels of information.
2. In database systems, data are instances of schemas whose structure is represented by instantiating metadata stored in the system catalog.

3. In software engineering, UML environments are designed on the basis of the MOF architecture [1] that also uses three levels of information.

A major problem that arises when managing MMS is the growing number of evolving metamodels. Coming back to our three examples, we notice that:

1. many ontology models are used: OWL [2], RDFS [3] or PLIB [4]. Moreover, most of these models are evolving regularly to satisfy new user needs;
2. each database system has its own system catalog structure supporting common features, with specificities not available in other system catalogs;
3. each UML environment supports a specific subset of the UML standard.

To support metamodels evolution, the OMG proposal consists in introducing a fourth upper level: the metametamodel [1]. Thus new metamodels are created as instances of this upper level. If this approach has been followed in software engineering and databases with systems such as ConceptBase [5] or Rondo[6] and languages such as mSQL [7] or QML [8], it is not generalized to ontology-based applications. These systems and languages could be used to represent ontologies but they would not exploit specific features of ontologies such as universal identification of concepts using namespaces/URI or multilingual descriptions.

The aim of this paper is to highlight capabilities of the OntoQL language for managing the various levels of information stored in a semantic MMS i.e, supporting semantic descriptions through ontologies. We have introduced this language in [9,10] by presenting its capabilities to query ontologies and instances with its formal definition. In this paper, we show that this language can be used to access and modify the metamodel level in addition to its capability to express semantic queries and its compatibility with SQL. To illustrate our proposals, we consider a concrete example of semantic MMS: databases embedding ontologies and their instances namely Ontology Based DataBases (OBDBs). But, the reader shall notice that this approach is generic. In particular, the core metamodel defined for designing the OntoQL language managing OBDBs can be set up for other MMSs. Indeed, as shown later in the paper, the structure as well as the semantics of this core metamodel can be extended according to the targeted application domain. We report several applications where this feature of OntoQL has been extensively and successfully used.

This paper is structured as follows. In the next section, we introduce an example showing the need of modifying the metamodel level in OBDBs, i.e. the ontology model. This example is used throughout this paper. In section 3, we present the OBDB context we have used to illustrate our approach. The capability of the OntoQL language to uniformly manipulate the three levels of an OBDB is then presented in section 4. Its implementation on the OntoDB OBDB [11] and application in several projects are described in section 5. Finally, our conclusion, as well as some challenges for future work, are given in section 6.

2 A Motivating Example

When an ontology must be designed, an ontology model is required. This choice is difficult not only because of the numerous existing ontology models but also because they all have their own particularities that may be simultaneously needed for the addressed problem. For example, in the context of the e-Wok Hub project[1] whose aim is to manage the memory of many engineering projects on the CO2 capture and storage, experts must design an ontology on this domain. They must choose an ontology model knowing that on the one hand they have to precisely and accurately represent the concepts of this domain (by defining their physical dimensions associated with their units of measure and the evaluation context). This need suggests to use an ontology model such as PLIB. On the other hand, constructors introduced by ontology models borrowed from Description Logics such as OWL are also required to perform inferences in order to improve the quality of documents search. This example shows the need for manipulating the ontology model used in order to take advantages of each model.

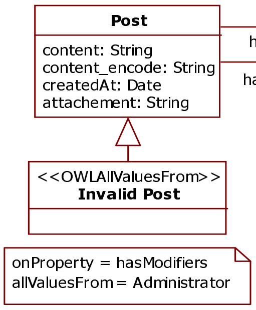


Fig. 1. An extract of an ontology for online communities

To illustrate the OntoQL language, we use an example of ontology. Instead of illustrating the use of OntoQL on a complex ontology related to the CO2 capture and storage domain, we have chosen an ontology example describing the on-line communities domain. This example, depicted in Figure 1, is influenced by the SIOC ontology (*http://sioc-project.org/*). This ontology defines concepts such as `Community`, `Usergroup` or `Forum`. In this example we have identified the `User` and `Post` classes specialized by the `Administrator` and `InvalidPost` classes. The `InvalidPost` class is defined as an OWL restriction (`OWLAllValuesFrom`) on the `hasModifiers` property whose values must be instances of the `Administrator` class. Thus, an invalid post is a post that may be modified by administrators only. Since the UML notation does not allow us to represent this constructor, we have used a stereotype and a note to represent the `InvalidPost` class. Also notice that the UML notation prevents us to show the whole classes and properties description of this ontology (e.g. synonymous names or associated documents).

Before developing our proposal, we overview some systems that support ontologies management within database systems.

[1] http://www-sop.inria.fr/acacia/project/ewok/index.html

3 Context of the Study: OBDBs

Our proposal has been designed to manage MMS. However, to provide a concrete example of the interest of the proposed language, we consider a particular type of MMS: databases that store both ontologies and their instances, namely OBDBs. This section gives an overview of existing OBDBs and their associated languages.

3.1 Different Storage Structures for OBDBs

The most simple schema to store ontologies in a database is a three columns table `(subject, predicate, object)` [12] inspired from RDF triples. In this representation all the information (both ontologies and instances) are encoded by triples. This representation supports ontology model storage and modification. However, the performances of this representation led the research community to propose new solutions [13].

The second representation approach for OBDB consists in separating ontologies from their instance data. Data are stored either by 1) a triple representation [14], 2) or by a vertical representation [14,15] where each class is represented by an unary table and each property by a binary table 3) or by a horizontal representation [11] where a table with a column for each property is associated to each class. Notice that whatever is the chosen representation for data, these systems use a fixed logical model depending on the used ontology model to store ontologies. By fixed, we mean a logical model that cannot evolve.

Finally, to our knowledge, OntoDB [11] is the only OBDB which separates ontologies and data and proposes a structure to store and to modify the underlying ontology model (PLIB). The representation of this part, namely the *metaschema*, allows a user to make the PLIB ontology model evolve or extend it with constructors from other ontology models if required.

As this section shows, the triple representation and the explicit representation of the metamodel are the only ways to enable the dynamic evolution of the ontology model used in an OBDB. In the next section we show the limits of existing languages for OBDBs to exploit this capability.

3.2 Existing Languages for OBDBs

Many languages have been proposed for managing OBDBs outlined in the previous section. This section overviews some of these languages according to the exploited ontology model.

Languages for RDFS

The RQL language [16] supports queries of both ontology and instances represented in RDFS. Many other languages with similar features have been proposed (see [17] for a survey). In RQL, all constructors of the RDFS ontology model are encoded as keywords in its grammar. For example the keyword `domain` is a function returning the domain of a property. As a consequence every evolution of this model requires a modification of the language grammar.

To manage the diversity of ontology models, the data model of the RQL language is not fully frozen. Indeed this data model can be extended by specializing the built-in entities `Class` and `Property`. However new entities can not be added except if they inherit from these two entities. As a consequence, PLIB constructors for associating documents to concepts of an ontology or the OWL constructor `Ontology` for grouping all the concepts defined in an ontology can not be represented. The data model of RQL can also be extended with new attributes. This extension may be useful to characterize concepts of an ontology (e.g., by a version number or an illustration). However such an extension requires to use the property constructor. So, the modification of the metamodel level also impacts a modification of the model level. Finally, even if these (partial) capabilities are offered by RQL, they are not supported or at least not explicitly on the OBDB RDF-Suite [15] and Sesame [14] which support this language.

Languages for RDF

SPARQL [18] is a recommendation of the W3C for exploiting RDF data. As a consequence it considers both ontologies and instances as triples. Thus, differing from RQL, SPARQL does not encode the constructors of a given ontology model as keywords of its grammar. This approach gives a total freedom to represent data, ontologies and the ontology model used as a set of triples. However, since ontology constructors such as class definition or subsumption relationship are not included in this model, SPARQL does not define any specific semantics for them. So SPARQL does not provide operators to compute the transitive closure of the subsumption relationship for given classes or operators to retrieve properties applicable on a class (those defined or inherited by this class). The result of a SPARQL query is dependent of the set of triples on which it is executed.

Langages independent of a given ontology model

To our knowledge, the SOQA-QL language [19] is the only language available to query ontologies and instances independently of the used ontology model. This capability has been provided by defining this language on a core ontology model (SOQA Ontology Meta Model) containing the main constructors of different ontology models. However this model is frozen. As a consequence evolutions of ontology models as well as addition of specific features are not supported.

As the previous analysis shows, each category of existing languages presents some limitations to support an uniform and shared manipulation of the three levels of an OBDB. These limits can be summarized as follows.

– Languages of the RQL category offer operators to exploit the semantics of the ontology model RDFS but extensions of this model are limited and mix the different levels of an OBDB.
– Langages of the SPARQL category offer a total freedom when manipulating the three levels of an OBDB considering all the information as triples. However, they do not offer operators supporting the exploitation of the usual semantics of ontology models.
– The SOQA-QL language supports the manipulation of data of an OBDB whatever is the used ontology model, provided that the used constructors

are in the core ontology model on which this language is defined. However constructors not included in this core model can not be added.

This analysis leads us to propose another language. We design this language, named OntoQL, to manage the three levels of a MMS encoded in an OBDB.

4 OntoQL: Managing the Three Levels of MMS

We first present the data model targeted by OntoQL. Then, we present the OntoQL operators that manage each information level.

4.1 Data Model

The OntoQL language shall fulfill two requirements (1) enabling the modification of the metamodel level and (2) ensuring that these modifications match the semantics of MMSs: an instance of a metamodel (level named M_2 in the MOF) defines a valid model (M_1) which is itself expected to represent populations instances (M_0). These two requirements are fulfilled in the following way. (1) Contrary to usual DBMS where the metamodel level is fixed, OntoQL assumes the availability of a metametamodel level which is itself fixed (M_3). So instantiating this metametamodel modifies the metamodel level ensuring the flexibility of the metamodel level. (2) The metamodel level is composed of a predefined core model associated to an operational semantics. We describe this core model in the next section and then we show how it can be extended.

Core Metamodel

In an OBDB, the metamodel corresponds to the ontology model used to define ontologies. Contrariwise to the SOQA-QL language, we have chosen to include in this core model only the shared constructors of the main ontology models used in our application domain (engineering): PLIB, RDFS and OWL. However, the possibility to extend this model remains available.

Figure 2 presents the main elements of this core ontology model as a simplified UML model. We name *entities* and *attributes* the classes and properties of this UML model. The main elements of this model are the following.

– An ontology (`Ontology`) introduces a domain of unique names also called *namespace* (`namespace`). It defines concepts which are classes and properties.
– A class (`Class`) is the abstract description of one or many similar objects. It has an identifier specific of the underlying system (`oid`) and an identifier independent of it (`code`). Its definition is composed of a textual part (`name`, `definition`) which can be defined in many natural languages (`MultilingualString`). These classes are organized in an acyclic hierarchy linked by a multiple inheritance relationship (`directSuperclasses`).
– Properties (`Property`) describe instances of a class. As classes, properties have an identifier and a textual part. Each property must be defined on the class of the instances it describes (`scope`). Each property has a range (`range)` to constraint its domain of values.

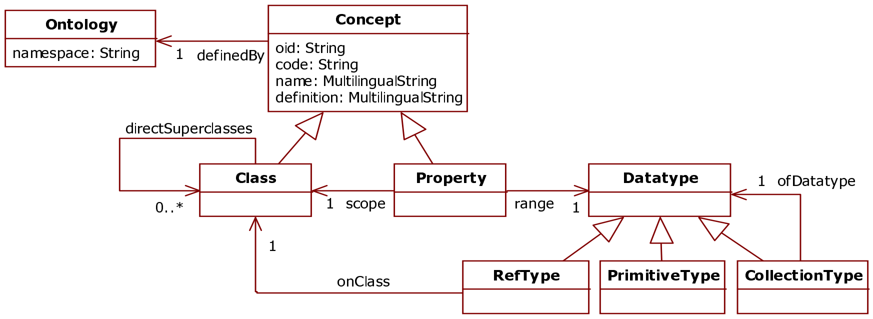


Fig. 2. Extract of the core metamodel

– The datatype (`Datatype`) of a property can be a simple type (`primitiveType`) such as integer or string. A property value can also be a reference to an instance of a class (`refType`). Finally, this value can be a collection whose elements are either of simple type or reference type (`collectionType`).

This core metamodel contains some specific features of ontologies (namespaces, multilinguism, universal identifier). A MMS based on this model supports ontology management and uses the defined ontologies to semantically describe other models. Moreover, since it includes the core components of standard metamodels (i.e, class, property and datatype), it may also be used as a kernel for numerous applications requiring a three-level architecture (e.g., enrichment of DBMS system catalog). Section 5 presents two examples issued from a real-world project.

Extensions of the Core Metamodel

The core metamodel can be extended with new entities and new attributes. When an extension is made by specialization, new entities inherit the behavior of their super entities. This behavior is defined in the operational semantics of the core metamodel. Thus, every specialization of the entity `Class` defines a new category of classes which supports by inheritance the usual behavior of a class. Every instance of the entity `Class` (or one of its specialization) defines a container which may be associated with instances. The name of this container is generated by a concretization function to access it through its representation at the meta level (M_2). Also every specialization of the entity `Property` defines relationships associating instances of classes to domain of values. The name of these relationships is also derived from the instances of properties which define them. Finally the specializations of the entity `Datatype` define domains of values that can be associated to properties.

4.2 OntoQL Operators for This Data Model

According to the data model defined, the creation of an element at the level M_i is done by instantiating the level M_{i+1}. However this process is not always practical. Indeed, in an usual DBMS, it would consist in creating a table using insert statements on the system catalog tables. So syntactic equivalences are systematically defined between insert statements (`INSERT`) at the level M_{i+1} and containers creation (`CREATE`) at the level M_i. These two syntactic constructions are valid but in general the second one is more compact. Figure 3 gives an overview of the statements available at the different levels as well as their meanings according to the target data structure. As shown on this figure, two equivalent syntactic constructions are provided for, on the one hand, inserting at the level M_3 or creating at the level M_2 and, on the other hand, inserting at the level M_2 or creating at the level M_1. In the following sections we describe more precisely the available statements.

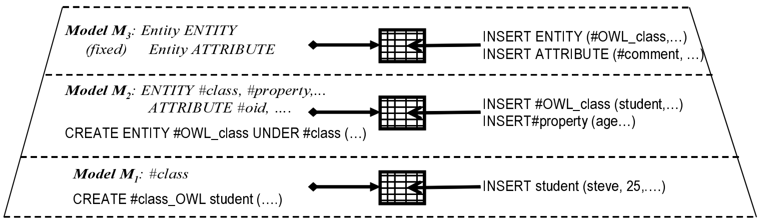

Model M3: Entity ENTITY
(fixed) Entity ATTRIBUTE
INSERT ENTITY (#OWL_class,...)
INSERT ATTRIBUTE (#comment, ...)
Model M2: ENTITY #class, #property,...
ATTRIBUTE #oid, ...
CREATE ENTITY #OWL_class UNDER #class (...)
INSERT #OWL_class (student,...)
INSERT#property (age...)
Model M1: #class
CREATE #class_OWL student (....)
INSERT student (steve, 25,....)

Fig. 3. Overview of the OntoQL language

Metamodel Definition Level and Model Manipulation/Querying

The OntoQL language provides operators to define, manipulate and query ontologies from the core metamodel presented in the previous section. Since this core metamodel is not static but can be extended, the elements of this level of information must not be encoded as keywords of the OntoQL language. So we have defined a syntactic convention to identify in the grammar a metamodel element. The convention we have chosen is to prefix each element of this model by the character `#`. This prefix indicates that this element definition must be inserted or searched in the metamodel level of the used MMS system.

To define the syntax of the OntoQL language, we have chosen to adapt SQL to the data model we have defined. Thus, the data definition language creates, modifies and deletes entities and attributes of the metamodel using a syntax similar to the manipulation of SQL user-defined types (`CREATE`, `ALTER`, `DROP`). In this paper, we give examples to demonstrate the use of OntoQL only. However a more formal definition (an algebra) can be found in [10].

Example. Add the `AllValuesFrom` constructor OWL to the core metamodel.

```
CREATE ENTITY #OWLAllValuesFrom UNDER #Class (
    #onProperty REF(#Property),
    #allValuesFrom REF(#Class) )
```

This statement adds the `OWLAllValuesFrom` entity to our core metamodel as subentity of the `Class` entity . This entity is created with two attributes, onProperty and allValuesFrom, which take respectively as value identifiers of properties and identifiers of classes.

To create, modify or delete the elements of ontologies we have defined a data manipulation language. In the same way as for the data definition language, we have adapted the data manipulation language of SQL (INSERT, UPDATE, DELETE) to the data model of this part.

Example. Create the class named InvalidPost of our example ontology.

```
INSERT INTO #OWLRestrictionAllValuesFrom
         (#name[en], #name[fr], #onProperty, #allValuesFrom)
  VALUES ('InvalidPost', 'Post invalide', 'hasModifiers', 'Post')
```

This example shows that values of multilingual attributes can be defined in different natural languages ([en] for English and [fr] for French). It also shows that the names of classes and properties can be used to identify them. Indeed the value of the onProperty attribute is hasModifiers: the name of a property (and not its identifier). Notice that thanks to the syntactic equivalences previously presented, we could have written this statement with the CREATE syntax closer to the creation of user-defined types in SQL. It is also often more concise because properties can be created in the same time.

Finally a query language allows users to search ontologies elements stored in the OBDB. By designing this language starting from SQL we benefit from the expressivity of the object-oriented operators introduced in this language by the SQL99 standard. Thus the OntoQL language provides powerfull operators such as path expressions, nesting/unnesting collections or aggregate operators. The following example shows a path expression.

Example. Search the restrictions defined on the hasModifiers property with the class in which the values of this property must be taken.

```
SELECT #name[en], #allValuesFrom.#name[en]
  FROM #OWLRestrictionAllValuesFrom
 WHERE #onProperty.#name[en] = 'hasModifiers'
```

This query consists in a selection and a projection. The selection retrieves the restrictions on the property named in English hasModifiers. The used path expression in this selection is composed of the onProperty attribute which retrieves the identifier of the property on which the restriction is defined and of the name attribute which retrieves the name in English of this property from its identifier. The projection also applies the name attribute to retrieve the name of the restriction and the path expression composed of the allValuesFrom and name attributes to retrieve the name of the class in which the property implied in the restriction must take its values.

Model Definition and Data Manipulation/Querying

At this level, OntoQL defines, manipulates and queries instances of ontologies. The corresponding OntoQL syntax has again been defined starting from SQL to get an uniform syntax and remain compatible with existing database models.

Thus each class can be associated to an *extent* which stores its instances. This data model generalizes the existing link between a user-defined type in SQL and a typed table which stores its instances. Indeed, contrary to a database schema which prescribes the attributes characterizing the instances of a user-defined type, an ontology only describes the properties that may be used to characterize the instances of a class. As a consequence the extent of class is only composed of the subset of properties which are really used to describe its instances. Thus, the OntoQL construct to create an extent of a class is similar to an SQL statement to create a typed table from a user-defined type. The only difference is that this statement precises the set of used properties.

Example. Create the extent of the `Administrator` class assuming that an administrator is described by his last name and his first name only.

```
CREATE EXTENT OF Administrator ("first name", "last name")
```

This statement creates a table with two columns to store instances of the `Administrator` class knowing that only the `first name` and `last name` properties describe them. This structure could also be a view on a set of normalized tables.

The semantics of OntoQL has been defined to search values of an instance for each property defined on its belonging class. When this property is not used to describe this instance, the `NULL` value is returned.

Example. Retrieve the users whose email address is known.

```
SELECT first_name, last_name FROM User WHERE email IS NOT NULL
```

The previous query will not return any administrator since the property `email` is not used on this class.

Another particularity of ontologies is that classes and properties have a namespace. The `USING NAMESPACE` clause of an OntoQL statement indicates in which namespace classes and properties must be searched. If this clause is not specified and no default namespace is available, OntoQL processes this clause as an SQL statement and thus it is compatible with SQL.

Example. Retrieve the content of the invalidate posts knowing that the class `InvalidPost` is defined is the namespace `http://www.lisi.ensma.fr`.

```
SELECT content FROM InvalidPost
USING NAMESPACE 'http://www.lisi.ensma.fr'
```

In this example the `InvalidPost` class and the `content` property are searched in the ontology which defines the namespace `http://www.lisi.ensma.fr`. In OntoQL many namespaces can be specified in the `USING NAMESPACE` clause and used to express queries composed of elements defined in different ontologies.

The next example illustrates a third particularity of ontologies: classes and properties have a textual definition that may be given in several different natural languages. This particularity is used in OntoQL to refer to each class and to each property by a name in a given natural language. The same query can be written in many natural languages.

Example. Retrieve the first and last names of users using a query in English (`7a`) and in French (`7b`).[2]

```
7a. SELECT "first name", "last name"  <=> 7b. SELECT prénom, nom
      FROM User                                    FROM Utilisateur
```

The query `7a` must be executed when the default language of OntoQL is set to English while the query `7b` requires the French default value.

Model and Data Querying

Queries on ontologies take as input/output parameter a collection of elements of the model level (e.g. classes or properties) while queries on instances take as input/output parameter a collection of instances of these elements (instances of classes and values of properties). The link between these two levels can be established in the two ways described in the next sections. (1) *From ontology to instances*, and (2) *from instances to ontology.*

From ontology to instances. Starting from classes retrieved by a query on ontologies, we can get the instances associated to these classes. Then we can express a query on data from this collection of instances. To query instances, following the approach of usual databases languages, OntoQL proposes to introduce an iterator `i` on the instances of a class `C` using the `C AS i` construct. The class `C` is known before executing the query. Moreover, OntoQL extends this mechanism with iterators on the instances of a class identified at run-time.

Example. Retrieve instances of the classes whose name ends by `Post`.

```
SELECT i.oid FROM #class AS C, C AS i WHERE C.#name[en] like
'%Post'
```

In this query, the `Post` and `InvalidPost` classes fulfill the condition of selection. As a consequence this query returns instances identifiers of these two classes. And since `InvalidPost` is a subclass of `Post`, instances identifiers of the `InvalidPost` class are returned twice.

From instances to Ontology. Starting from instances retrieved by a query on data, we can retrieve the classes they belong to. Then a query on ontologies starting from this collection of classes can be expressed. OntoQL proposes the operator `typeOf` to retrieve the *basic class* of an instance, i.e. the minorant class for the subsumption relationship of the classes it belongs to.

Example. Retrieve the English name of the basic class of `User` instances.

```
SELECT typeOf(u).#name[en] FROM User AS u
```

This query iterates on the instances of the `User` class and on those of the `Administrator` class as well. For each instance, the query returns `User` or `Administrator` according to its member class.

[2] In the following examples, the default namespace is `http://www.lisi.ensma.fr`

4.3 Exploitation of the Capability of the OntoQL Language

Building Defined Concepts or Derived Classes

OntoQL can be used to compute the extent of a *defined concept* i.e., a concept defined by necessary and sufficient conditions in terms of other concepts. It also corresponds to a derived class in UML terminology.

Example. Build the extent of the class `InvalidPost`.

```
CREATE VIEW OF InvalidPost AS
SELECT p.* FROM Post AS p WHERE NOT EXISTS
  (SELECT m.* FROM UNNEST(p.hasModifiers) AS m
    WHERE m IS NOT OF REF(Administrator))
```

This query iterates on the instances `p` of the `Post` class. For each instance a subquery determines if this post has been modified only by administrators using the operator `UNNEST` (unnest a collection) and `IS OF` (type testing).

Model Transformation

The OntoQL language can also be used for model transformation. For example, an ontology represented in a source ontology model can be transformed in another one in a target ontology model. It requires to encode with the OntoQL operators, the defined equivalences between the source and target models.

Example. Transform the OWL classes into PLIB classes knowing that the attribute `comment` in OWL is equivalent to the attribute `remark` of PLIB.

```
INSERT INTO #PLIBClass (#oid, #code, #name[fr], #name[en], #remark)
SELECT (#oid, #code, #name[fr], #name[en], #comment) FROM #OWLClass
```

This statement creates a PLIB class for each OWL class. The value of the `remark` attribute of the created PLIB classes is defined with the value of the `comment` attribute of the OWL source classes. Other complex transformations may be written within this approach. We are currently extending OntoQL to allow a complete transformation language.

5 Implementation and Cases Study

OntoQL is implemented on the OntoDB OBDB. We briefly describe this implementation in next section, and then overview two case studies where this implementation has been put into practice.

5.1 Implementation of the OntoQL Engine

Two approaches could be followed to implement the OntoQL language on an OBDB. The first method consists in translating OntoQL into SQL. This approach has been followed in the RDF-Suite [15] system. The second method, followed by Sesame [14], consists in translating an OntoQL statement into a set of API-functions call that return a set of data of the OBDB.

The first solution benefits from the important effort on SQL optimization. However it has the drawback to be dependent of the OBDB on which the language is implemented. The second solution ensures the portability of the implementation on different OBDBs since it is possible to provide different implementation of the API for each OBDB. However in this case optimization is delegated to the implemented query engine.

In our implementation we have mixed these two methods. OntoQL is translated into SQL using an API which encapsulates the specificities of the OBDB. It follows 3 main steps: (1) generation of OntoQL algebra tree from a textual query (2) transformation of this tree into a relational algebra tree using an API (3) generation of SQL queries from this tree.

In addition to the implementation of the OntoQL Engine we have developed the following tools to ease the exploitation of this language.

- *OntoQL*Plus*: an editor of OntoQL statements similar to SQL*Plus provided by Oracle or isql provided by SQLServer.
- *OntoQBE*: a graphical OntoQL interface that extends the QBE interface such as the one provided by Access.
- *JOBDBC*: an extension of the JDBC API to execute OntoQL statements from the JAVA programming language.

Demonstrations and snapshots of these tools are available at `http://www.plib.ensma.fr/plib/demos/ontodb/index.html`

5.2 Case Study: CO2 Capture and Storage

The capabilities of the OntoQL language to manipulate the metamodel level have been exploited in various projects [20,21,22] and especially in the eWokHub project we have introduced in section 2. Two extensions we have realized are outline below: (1) annotation of engineering models and (2) handling user preferences.

Engineering Models Annotation

The CO2 capture and storage rely on various engineering models. Engineers have to face several interpretation difficulties due to the heterogeneity of these models. To ease this process, we have proposed to annotate these models with concepts of ontologies [20]. However, the notions of annotations and engineering models were not available at the metamodel level. Thus, OntoQL was used to introduce these notions as first-order model concepts using a stepwise methodology. First, elements of the engineering models were created with the `CREATE ENTITY` operator. Then, an association table was defined to annotate the engineering models by a class of an ontology. Once the metamodel was extended, OntoQL has been used to query the engineering models departing from the ontology concepts.

User Preferences Handling

When the amount of ontological data (or instances) available becomes huge, queries return a lot of results that must be sorted by a user in order to find the relevant ones. This requirement raised from the eWokHub project where a huge amount of documents and engineering models were annotated by concepts and/or instances of ontologies. As a solution, we have enriched the metamodel to handle user preferences when querying the OBDB. Our proposition is based on a model of user preferences [21] defined at the metamodel level and stored in the OBDB using the `CREATE ENTITY` operator of OntoQL. This preference model is linked to the ontology model by associating preferences to classes or properties of ontologies (`ALTER ENTITY`). Finally, OntoQL has been extended with a `PREFERRING` clause interpreting preferences when querying the OBDB.

6 Conclusion

This paper presents the OntoQL language that combines capabilities of metamodeling languages i.e, a uniform manipulation of the three levels composing MMS systems with those of ontology query languages i.e, queries at the knowledge (semantic) level. This language is based on a core metamodel containing the common and shared constructors of most of the usual ontology models. Thus this core metamodel is used to define ontologies. Moreover, it can be extended in order to take into account specific features of a particular metamodel as shown in section 5. An implementation of the ontoQL language has been developed on top of the OntoDB ontology based database together with a suite of tools similar to those existing for SQL. This language has been used in the application domain of the CO2 capture and storage to annotate engineering models, to handle preferences and to semantically index Web Services [22].

Currently, we are studying new mechanisms to extend the core metamodel with new operators and behaviors. The challenge is to enable the automatic definition of operators behavior without any interactive programming but by exploiting a metamodel for behavioral modeling. More precisely, we are working on the possibility to dynamically add new functions that could be triggered during query processing preserving persistance of the models and their instances.

References

1. Object Management Group: Meta Object Facility (MOF), formal/02-04-03 (2002)
2. Dean, M., Schreiber, G.: OWL Web Ontology Language Reference. W3C Recommendation February 10 (2004)
3. Brickley, D., Guha, R.V.: RDF Vocabulary Description Language 1.0: RDF Schema. World Wide Web Consortium (2004)
4. Pierra, G.: Context Representation in Domain Ontologies and its Use for Semantic Integration of Data. Journal Of Data Semantics (JODS) X, 34–43 (2007)
5. Jeusfeld, M.A., Jarke, M., Mylopoulos, J.: Metamodeling for Method Engineering. MIT Press, Cambridge (2009)

6. Melnik, S., Rahm, E., Bernstein, P.A.: Rondo: a programming platform for generic model management. In: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data (SIGMOD 2003), pp. 193–204 (2003)
7. Petrov, I., Nemes, G.: A Query Language for MOF Repository Systems. In: Proceedings of the OTM 2008 Conferences (CoopIS 2008), pp. 354–373 (2008)
8. Kotopoulos, G., Kazasis, F.C.S.: Querying MOF Repositories: The Design and Implementation of the Query Metamodel Language (QML). In: Digital EcoSystems and Technologies Conference (DEST 2007), pp. 373–378 (2007)
9. Jean, S., Aït-Ameur, Y., Pierra, G.: Querying Ontology Based Database Using OntoQL (an Ontology Query Language). In: Proceedings of Ontologies, Databases, and Applications of Semantics (ODBASE 2006), pp. 704–721 (2006)
10. Jean, S., Aït-Ameur, Y., Pierra, G.: An Object-Oriented Based Algebra for Ontologies and their Instances. In: Ioannidis, Y., Novikov, B., Rachev, B. (eds.) ADBIS 2007. LNCS, vol. 4690, pp. 141–156. Springer, Heidelberg (2007)
11. Dehainsala, H., Pierra, G., Bellatreche, L.: Ontodb: An ontology-based database for data intensive applications. In: Kotagiri, R., Radha Krishna, P., Mohania, M., Nantajeewarawat, E. (eds.) DASFAA 2007. LNCS, vol. 4443, pp. 497–508. Springer, Heidelberg (2007)
12. Harris, S., Gibbins, N.: 3store: Efficient bulk rdf storage. In: Proceedings of the 1st International Workshop on Practical and Scalable Semantic Systems (2003)
13. Theoharis, Y., Christophides, V., Karvounarakis, G.: Benchmarking Database Representations of RDF/S Stores. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 685–701. Springer, Heidelberg (2005)
14. Broekstra, J., Kampman, A., van Harmelen, F.: Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. In: Horrocks, I., Hendler, J. (eds.) ISWC 2002. LNCS, vol. 2342, pp. 54–68. Springer, Heidelberg (2002)
15. Alexaki, S., Christophides, V., Karvounarakis, G., Plexousakis, D., Tolle, K.: The ics-forth rdfsuite: Managing voluminous rdf description bases. In: Proceedings of the Second International Workshop on the Semantic Web, SemWeb 2001 (2001)
16. Karvounarakis, G., Magkanaraki, A., Alexaki, S., Christophides, V., Plexousakis, D., Scholl, M., Tolle, K.: Querying the Semantic Web with RQL. Computer Networks 42(5), 617–640 (2003)
17. Bailey, J., Bry, F., Furche, T., Schaffert, S.: Web and Semantic Web Query Languages: A Survey. In: Reasoning Web, pp. 35–133 (2005)
18. Prud'hommeaux, E., Seaborne, A.: SPARQL Query Language for RDF. W3C Recommendation (January 15, 2008)
19. Ziegler, P., Sturm, C., Dittrich, K.R.: Unified Querying of Ontology Languages with the SIRUP Ontology Query API. In: Proceedings of Business, Technologie und Web (BTW 2005), pp. 325–344 (2005)
20. Mastella, L.S., Aït-Ameur, Y., Jean, S., Perrin, M., Rainaud, J.F.: Semantic exploitation of persistent metadata in engineering models: application to geological models. In: Proceedings of the IEEE International Conference on Research Challenges in Information Science (RCIS 2009), pp. 147–156 (2009)
21. Tapucu, D., Diallo, G., Aït-Ameur, Y., Ünalir, M.O.: Ontology-based database approach for handling preferences. In: Data Warehousing Design and Advanced Engineering Applications: Methods for Complex Construction, pp. 248–271 (2009)
22. Belaid, N., Ait-Ameur, Y., Rainaud, J.F.: A semantic handling of geological modeling workflows. In: International ACM Conference on Management of Emergent Digital EcoSystems (MEDES 2009), pp. 83–90 (2009)

A Framework for OLAP Content Personalization

Houssem Jerbi, Franck Ravat, Olivier Teste, and Gilles Zurfluh

IRIT, Institut de Recherche en Informatique de Toulouse
118 route de Narbonne, F-31062 Toulouse, France
`{jerbi,ravat,teste,zurfluh}@irit.fr`

Abstract. A perennial challenge faced by many organizations is the management of their increasingly large multidimensional databases (MDB) that can contain millions of data instances. The problem is exacerbated by the diversity of the users' specific needs. Personalization of MDB content according to how well they match user's preferences becomes an effective approach to make the right information available to the right user under the right analysis context. In this paper, we propose a framework called OLAP Content Personalization (*OCP*) that aims at deriving a personalized content of a MDB based on user preferences. At query time, the system enhances the query with related user preferences in order to simulate its performance upon an individual content. We discuss results of experimentation with a prototype for content personalization.

Keywords: OLAP content, Multidimensional databases, Personalization, Preferences.

1 Introduction

Personalized data selection aims at providing users with only relevant data from the huge amount of available information. It has become an increasingly important problem for many applications in a variety of fields [12,17]. However, it has not been well-studied in the context of data warehousing, including many important applications like business intelligence, OLAP (On-Line Analytical Processing), and decision support.

1.1 Context and Issues

The MultiDimensional Database (MDB) design process aims at defining a unified multidimensional schema which considers the analysis needs of a group of users. Typically a MDB schema is modeled through a constellation of subjects of analysis, called facts, and axes of analysis, called dimensions [13]. An ETL (Extract, Transform, and Load) process takes charge of extracting the data from different source systems. These data, called the MDB *content*, represent the schema instances, *i.e.*, the fact data as well as the dimensions parameters instances. Typically, a single ETL process is set up for a MDB because designing such process is extremely complex and time consuming [16]. However, users sharing the same MDB have usually different perceptions of its content since they have various interests and goals [3, 15]. For example, a user is interested in analyzing sales only in France, whereas another user likes to see more global data of all the European countries. Furthermore, the same user may have interest on data content that varies from one analysis context to another [3, 10]. For example, a decision-maker needs to analyze data about all years in the context of analysis of purchases, while she considers only the current year when analyzing sales. Therefore, users must often define complex queries with several constraints to restore accurate data, which could be a hard task.

Besides, the data loading process may consolidate a quite large content as it may deliver information to many kinds of users. The increasing amount of data stored in OLAP databases often leads to the many-answers problem. Therefore, acquiring the required information is more costly than expected. In this context, the research agenda in [15] identifies personalization as one of the main topics to be addressed by both academics and practitioners in order to avoid information flooding in the OLAP context.

We argue that personalization of MDB content represents an important step to provide customized content for every particular user, thus better satisfying users.

1.2 Motivating Example

In this paper, we use for our running example a multidimensional database housing the laboratory publications and information about the researchers' missions. Part of the schema is illustrated in Fig. 1. Publications and research missions can be analyzed by *events, authors and dates*. Within the dimension *events*, the attributes *type*, *category*, *editor*, and *level* refer to the event type (conference, journal, or workshop), category of the publication (IEEE, ACM, …), editor of the proceeding, and the event level (international or national), respectively.

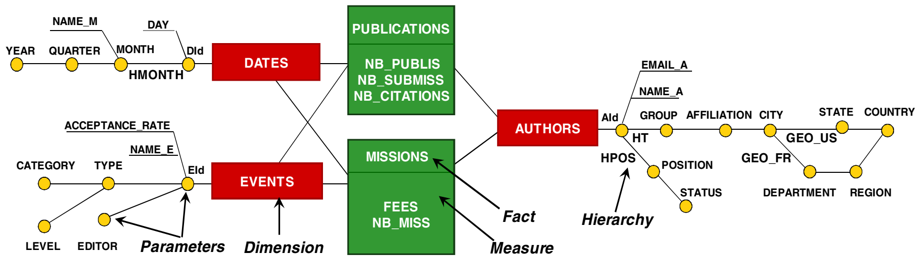


Fig. 1. Example of MDB schema

Consider for instance two research teams' heads. Prof. Brian is mainly interested in publications in IEEE and ACM events, while Prof. John is rather interested in all publications. In order to analyze the research performance of their teams, the decision-makers need to observe the number of publications during the current year by quarter and by category of events. When asking the laboratory secretary about such information, she provides each decision-maker with a dashboard according to his interests and specific constraints. She relies upon the personal relationship she has with everyone and her past informal communication with him. However, when applying a query under a decision-support tool, the query engine restores for both of decision-makers the same answer including data of all events categories. Therefore, Prof. Brian is forced to explore the whole multidimensional result space to find data about IEEE and ACM events which is a frustrating task. Note that the publication year (year = 2010) represents for Prof. Brian an immediate need which is explicitly expressed within his query, while the publication category is a long-term information representing a main interest that distinguishes him from the other researchers. Such feature is a constraint that the OLAP system must fulfill to satisfy the user expectations.

We argue that the MDB system will restore more personalized data by better understanding of the user specific interests. For this purpose, user analysis preferences should be stored in a user profile, then, the system could automatically integrate them into the original query, in order to satisfy expectations of each user.

1.3 Aims and Contributions

In this paper, we take a step towards personalized content access in OLAP systems. We present a new framework, called OLAP Content Personalization (*OCP)*, which aims at personalizing dynamically the MDB content with the use of user profiles. The main contributions of the paper are the following:

- *OCP* framework. The main goal of *OCP* is defined as follows. Given a set of users that share a MDB, we wish to extract a personalized content for each user. In practice, this leads to include the user profile parts into the qualification of the user queries to further restrict the universe of data content that generates the query result. The major steps for personalized content extraction are: (a) preference selection, where the preferences relevant to the query and most interesting to the user are derived from the user profile, and (b) preference integration, where the derived preferences are integrated into the user query producing a modified personalized one, which is actually executed.
- *Preference model for user profiles.* User preferences are of the form (*predicate*, *degree*) | *X*, meaning that the user is interested about data issued from *predicate* in the analysis context *X*. Each preference is associated with a degree of interest. Preferences are stored in a user profile as long-term information need.
- *Experimental results.* The proposed framework has been implemented and it is discussed through a set of experiments that show its potential.

Roadmap. The rest of the paper is organized as follows: Section 2 describes the personalized data model. Section 3 presents our personalization framework. Section 4 discusses results of experimentation. Section 5 presents related work. Finally, conclusions and future work are drawn in the last section.

2 Personalized OLAP Data

We define OLAP content personalization like the process of adapting the MDB content to the user specific interests. This leads to enrich the MDB by a descriptor layer consisting of the user-related information needed to personalize. Such information is specified in a so-called user profiles.

2.1 OLAP Data Modeling

OLAP data are designed according to a constellation of facts and dimensions. Dimensions are usually organized as hierarchies, supporting different levels of data aggregation.

A *constellation* regroups several facts, which are studied according to several dimensions. It is defined as (N^{CS}, F^{CS}, D^{CS}, $Star^{CS}$) where N^{CS} is the constellation name, F^{CS} is a set of facts, D^{CS} is a set of dimensions, $Star^{CS}$: $F^{CS} \rightarrow 2^{D^{CS}}$ associates each fact to its linked dimensions.

A *fact*, noted $F_i \in F^{CS}$, reflects information that has to be analyzed through indicators, called measures. It is defined as (N^{Fi}, M^{Fi}) where N^{Fi} is the fact name, $M^{Fi}=\{f_1(m_1),\ldots,f_w(m_w)\}$ is a set of *measures* associated to aggregation functions f_i.

A *dimension,* noted $D_i \in D^{CS}$, is defined as (N^{Di}, A^{Di}, H^{Di}) where N^{Di} is the dimension name, $A^{Di} = \{a^{Di}_1,\ldots, a^{Di}_u\}$ is a set of dimension attributes, $H^{Di} = \{H^{Di}_1,\ldots, H^{Di}_v\}$ is a set of hierarchies. Within a dimension, attribute values represent several data granularities according to which measures could be analyzed. In a same dimension, attributes may be organized according to one or several hierarchies.

A *hierarchy*, noted $H^{Di}_j \in H^{Di}$, is defined as (N^{Hj}, P^{Hj}) where N^{Hj} is the hierarchy name, $P^{Hj}=<id^{Di}, p^{Hj}_1,\ldots, p^{Hj}_{vj}, All>$ is an ordered set of attributes, called *parameters*, which represent useful graduations along the dimension D_i, $\forall k \in [1..v_j]$, $p^{Hj}_k \in A^{Di}$.

The constellation depicted in Fig. 1 consists of two facts (*publications* and *missions*) and three dimensions (*events*, *dates* and *authors*). Within the dimension *authors*, the hierarchy *GEO_FR* allows analyzing data according to the French geography (*i.e.*, cities are regrouped into departments and regions), while *GEO_US* hierarchy illustrate the US geography, *i.e.*, cities are regrouped into states.

2.2 User Profiles Modeling

Given our focus on personalization of OLAP content access, our user model assigns preferences to the MDB content. More specifically, we model OLAP user's preferences by 1) preferences on dimension data (*e.g.*, publications during the last 3 years), and 2) preferences on fact data (*e.g.*, a number of publications greater than 10).

Definition. Given a MDB, for an attribute *A* associated with a data type Type(A), a preference P^A is defined as ($pred^A$; θ), where

- $pred^A$ is a disjunction of restriction predicates of the form $A\ op\ a_i$ that specify condition on the values $a_i \in$ Type(A),
- θ is a real number between 0 and 1 that indicates the user interest degree in results that exactly satisfy this restriction predicate.

According to *A*, the predicate *pred* may be a restriction of fact data, *i.e.*, A is a measure associated with an aggregate function $f_i(m^F_i) \in M^F$, or a condition on dimension data, *i.e.*, A is a dimension attribute $p_i \in A^{Di}$. We assume op $\in \{=,<,>,\leq,\geq,\neq\}$ for numerical attributes and op $\in \{=,\neq\}$ for the other data types.

The level of a user's desire to analyze contents may be different depending on her ongoing analysis context. For example, a user can denote different preferences when she is focusing on publication statistics of her team than when she is analyzing those of all researchers, in general. We argue that the preference model should allow handling multiple ratings in different analysis contexts. For this reason, a user preference may be associated with a specific analysis context. Such context is more or less general.

Preference analysis context, or simply *preference context (cp)*, is of the form ([F_i (.$f_i(m^{Fi})$ / pr_1^{mi}/…/ $pr_k^{mi})^*$]; [D_1 / (a_j^{D1}/pr_1^{aj}/…/ $pr_s^{aj})^*$] ; … ; [D_p/(a_k^{Dp}/pr_1^{ak}/…/ $pr_t^{ak})^*$])[1], where

- F_i is the analyzed subject through a set of measures $m_j \in M^F$ associated with aggregate functions f_j (AVG, SUM, …),
- D_1,…, D_p are the analysis dimensions, $D_j \in D^{CS}$, $a_j^{Dq} \in A^{Dq}$ is the detail level within D_q, and
- pr_y^x is a restriction predicate over the values of an attribute or a measure.

Certain elements of *cp* may be empty. Actually, the context of a user preference does not necessarily contain all the analysis components. If there is no descriptor for a context element, the value *ALL* is assumed. For instance, a preference related to *cp* = (*ALL*, *Authors/position*='PhD Student', *Dates*/*ALL*) is relevant for any analysis of data related to PhD students according to the temporal axis, irrespective of the analysis subject and the detail level within dimension Dates.

To cope with the context-awareness of user preferences, a preference P_i is mapped to an analysis context cp_j in order to specify its applicable circumstances. This leads to contextual mappings m_{ij} of the form (P_i, cp_j). The semantics of such mappings in terms of the MDB content is the following: for m_{ij} = (($pred_i$, θ_i), cp_j), θ_i expresses the importance of taking into account the content involved by $pred_i$ into the qualification of the analysis context cp_j.

Note that some preferences are not context-aware; they are *global preferences* that hold in any analysis context. By convention, global preferences are mapped to '*ALL*'.

User preferences and associated contexts form a user profile that is used at query time to expeditiously provide personalized content.

A user profile P_u is defined as a set *M* of contextual mappings which relates user preferences to a set of analysis contexts *CP*: $P_u = \{m_{ij}(P_i,cp_j) \mid P_i = (pred_i,\theta_i), cp_j \in CP\}$.

Example 1. In addition to his interest in IEEE and ACM events, Prof. Brian has the following contextual preferences: (Status = 'permanent'; 0.9), (SUM(Nb_publis) >1; 0.6), (Affiliation = 'IRIT'; 0.5), (Affiliation = 'IRIT'; 0.9), (SUM(Fees) ≥ 50Eur; 0.8), and (AVG(Fees) ≥ 200Eur; 0.75). These preferences are associated to contexts:

- cp_1: analysis of publications by event; cp_1 = Publications; Events
- cp_2: analysis of the publications number during the current year; cp_2 = Publications.Sum(Nb_publis); Dates/Year = 2010
- cp_3: analysis of data of PhD students; cp_3 = Authors/ Position = 'PhD student'

Fig. 2 depicts Prof. Brian profile. Several preferences may be related to the same attribute (*e.g. Fees*). A preference predicate can be related to several contexts, eventually with different scores (*e.g. Affiliation='IRIT'*). Likewise, an analysis context may be mapped to many preferences (*e.g.* cp_2).

[1] The symbol "*" denotes zero or more repetitions of elements.

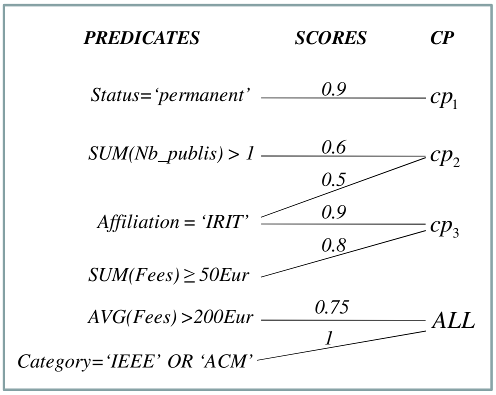


Fig. 2. Prof. Brian profile

Note that preferences may evolve through time. Thus, Fig. 2 illustrates Prof. Brian profile for a given point in time. The content personalization process (see section 3) is not affected by changes in the profiles, since it considers the stored preferences at runtime.

3 *OCP* Framework

The general architecture of an *OCP* framework is depicted in Fig. 3 and includes several modules surrounding a traditional OLAP content access module. The system keeps a repository of user-related information (*user profiles*) that is either inserted explicitly or inferred within the user log data. This profile information is integrated into an incoming request during content selection (*query reformulation*). The major steps for query reformulation are: (a) extraction of user profile parts that are related to the query (*profile manager*), then (b) construction of a Personalized Virtual Content, and query enhancement (*PVC constructor*).

This paper concentrates on taking advantage of user profiles for content access personalization in OLAP. Profiles generation is out of the scope of this paper.

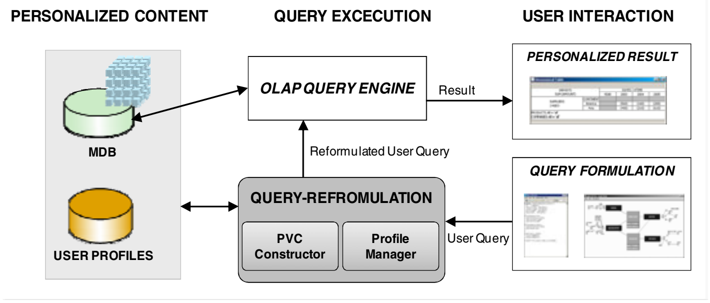


Fig. 3. Framework overview

3.1 Preferences Engineering

Active preferences. Some preferences among those stored in the user profile are related to the incoming query *Q*. Without loss of generality, we focus on OLAP queries that request data from one fact; *i.e.*, a query is applied on a star schema *S* of the MDB. Therefore, only preferences that are related to *S* are considered to execute *Q*, *i.e.*, preferences defined on the fact F^Q and the measures $M^Q_1,\ldots, M^Q_w$ of the user query, as well as all dimensions that are connected to F^Q, *i.e.* $Star^{CS}(F^Q)$. Such preferences are called *active preferences.*

In order to speed up the active preferences extraction, an offline preprocessing step consists in storing within a metatable mappings between couples (fact, measure) of the MDB and the related preferences. For a given query *Q*, active preferences are those mapped to (F^Q, M^Q).

Example 2. Suppose that Prof. Brian poses the query Q_1: number of publications during the two last years by event type and by research group. Preferences (SUM(Fees) ≥ 50Eur; 0.8) and (AVG(Fees) ≥ 200Eur; 0.75) are not active w.r.t. Q_1 since they are related to the measure *Fees* which is not concerned by Q_1.

Candidate preferences. As user preferences are contextualized, the content personalization considers only those active preferences that are associated with the current analysis context. Such preferences form, in addition to the active global preferences, the set of candidate preferences P^{Cand}.

Definition. The Current Analysis Context *CAC* is the context of analysis induced by the user query *Q*. It consists of the fact F^Q, measures $M^Q_1,\ldots, M^Q_w$, dimensions d^Q_1, ..., d^Q_x, the detail levels p^Q_1, ..., p^Q_y, and possibly restrictions $pred^Q_1$, …, $pred^Q_z$ of the query Q.

P^{Cand} is determined by context resolution where preferences that are associated with *CAC* are selected. This step is described in section 3.2.

However, during the *OCP* process, conflicts may arise.

Conflicts Resolution. We identify two types of conflicts: 1) a preference is conflicting with a query if it is conflicting with a condition already there; and 2) a preference is conflicting with another preference if their predicates are contradictory.

Definition. We say that two predicates $pred_1$ and $pred_2$ (normalized predicates, *i.e.*, conjunction of disjunctions) are conflicting, if $pred_1 \wedge pred_2 = False$ (*i.e.*, their conjunction returns no results).

Conflicts are detected syntactically with the use of the predicate logic rules. However, syntactical resolution does not identify some conflicts that arise at the semantic level. For example, a query condition *Name_E = 'CIDR'* (such that CIDR is a biennial conference that holds at odd-numbered years) is conflicting with the preference *Year = 2010*. In order to decide whether a preference is conflicting with a query at the semantic level, additional knowledge about the data is needed other than information derived from the constellation.

Each active preference is validated with the remaining active ones, then with all the conjunctive combinations of the user query predicates. Actually, although certain preferences are not conflicting with each query predicate individually, its conjunction with all of them may return no results. When a preference predicate is conflicting with a query predicate, the former is rejected. Besides, conflicts between preferences are resolved either offline or online. In the first case, conflict arises when a new preference P_1 is defined on the same target (measure or dimension attribute) than another preference P_2 that already exists in the user profile: P_1 and P_2 are either global preferences or contextual preferences associated with the same context. We assume that the most recent preference is maintained. However, some conflicts are detected at query time during the context resolution: a contextual preference may be conflicting with a global one; moreover, two contextual preferences may be conflicting if they are associated with different contexts, and they are both relevant for the user query. In this case, both preferences are discarded.

The output of the conflict resolution step is a set of *homogenous* candidate preferences.

3.2 Preference Selection

This step consists in identifying at runtime user preferences that are related to the current analysis context involved by the user query. Intuitively, only preferences that are associated with *CAC* (P_i | cp_i = *CAC*) are considered. However, some preferences related to a subset of the *CAC* are also useful.

Example 3. Consider Prof. Brian query Q_1 (see Example 2). Prof. Brian has active contextual preferences associated with contexts cp_1, cp_2, and cp_3. Which of them to use?

Preference associated with cp_1 is valid since *CAC* concerns also the analysis of publications by event. Likewise, those related to cp_2 are relevant as the current year is among the two last years that are concerned by *CAC*. However, the active preference related to cp_3 has to be discarded because the analysis precision of *CAC* through the axis *authors* is different from the attribute *position*. Thus, a candidate preference context must consist of items that already exist in *CAC*, and the common items must be organized according to the same hierarchical order in *CAC* (*e.g.,* analyzing publications by years then by months is different form analyzing publications by months then by years). Overall, cp_i of a candidate preference is *more general* than (or *dominates*) *CAC*.

Context Tree. In order to formalize the context resolution paradigm, we need to define a tree representation for *CAC* and preference contexts cp_i to state which context preference is close to *CAC*. The context tree reflects the nature of the relationships between components of an OLAP analysis [10]. There are two types of nodes in this tree:

- *structure nodes*, for the analysis context fact, measures, dimensions, and attributes
- *predicate nodes,* one for each restriction predicate of the analysis context

All predicates on the same dimension attribute (respectively measure) are conjunctively combined and represented by one node that is linked to the underlying attribute node (resp. measure node). Before building the preference context tree and the *CAC* tree (*BuildTreeC*), repeated and transitive predicates are removed. This allows avoiding useless matching and further inaccurate estimation of the tree size.

Context Matching. Intuitively, it is possible to state that an analysis context C_1 (represented by means of the tree T_1) is more general than another context C_2 (represented by T_2), if T_1 is included in or equal to T_2 ($T_1 \subseteq T_2$). This is determined through context trees matching that checks if all edges of T_1 belong to T_2.

Context matching is performed by traversing synchronously the two context trees in a breadth-first order, *i.e.*, level by level, as they are not so deep. For each iteration, the system checks if the two corresponding nodes from the two trees respectively fit. This leads to nodes comparing: two structure nodes fit together if they consist of the same node value, while predicate nodes comparison is performed by predicate logic rules to check if one predicate is either a reduced form of the second, or is equal to the second.

The context matching generates the set of preference contexts that dominate *CAC*. Contexts that overlap with *CAC* are rejected, since integrating their related preferences leads to approximate results that correspond to contexts near to *CAC*.

Example 4. Let's revisit the previous example. cp_3 is rejected since its tree overlaps with the *CAC* tree. Fig. 4 depicts the tree of cp_2 and its matching with the tree of *CAC*.

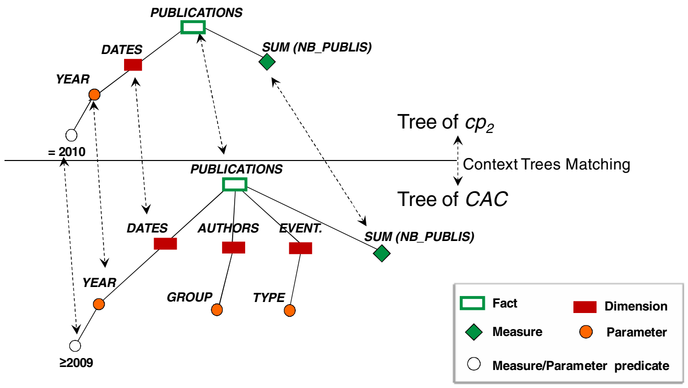


Fig. 4. Example of context trees matching

Preference Selection Algorithm. Preference selection consists in extracting preferences stored in user profile, that, when combined with the user query, would maximize the interest in the results. Intuitively, the more preferences are considered, the more user interest in the query result is. In practice, this may lead to impractical implantation, since maximum interest is achieved by incorporating all preferences, and the resulting query is likely to be very expensive or have an empty answer. In order to cope with such personalization problems, the system may select top K preferences. The parameter K may be retrieved from the user profile based on information collected by the system. Alternatively, it may be automatically derived at query time considering various constraints, *e.g.*, the query execution time of the final query, the query result size, etc.

Thereafter, the top K preferences will be integrated into the qualification of *CAC*. Thus, candidate preferences P_i are ranked according to their interest degrees under *CAC*, called $score(Pi)^{CAC}$. As $cp_i \subseteq CAC$, $score(Pi)^{CAC}$ should be a function of θ_i, which is the degree of interest of P_i under cp_i. In principle, one may imagine several such functions. Anyone of them, however, should satisfy the following condition, in order to be intuitive: $score(Pi)^{CAC} = f(\theta_i) \leq \theta_i$. In other words, the degree of interest in a preference decreases as it is related to a more general context. In this way, the more the context of a given preference covers CAC, the more its related score is preserved. Thus, $score\ (P_i)^{CAC}$ is proportional to the covering rate of the context of P_i. The formal definition is the following:

$$Score\ (P_i)^{CAC} = \theta_i * \text{covering\_rate}_{CAC}^{cpi} \tag{1}$$

$$\text{covering\_rate}_{CAC}^{cpi} = \frac{Card(cpi)}{Card(CAC)} \in [0, 1] \tag{2}$$

$Card(cp_i)$ (resp. $Card(CAC)$) is the cardinality (*i.e.*, number of edges) of the cp_i tree (resp. CAC tree); hence $\text{covering\_rate}_{CAC}^{cpi}$ is a real between 0 and 1, since $cp_i \subseteq CAC$. Therefore, $score(Pi)^{CAC}$ essentially approaches θ_i as more and more cp_i covers CAC. Note that for global preferences, $score(Pi)^{CAC} = \theta_i$, since they are relevant under every analysis context.

```
Input :
CAC: the current analysis context induced by the user query Q
$P^{ACT}$ = {($P_i$,$cp_i$), …, ($P_m$,, $cp_m$)}: user active preferences
K= number of preferences to select
Output : $P^{CAND}$ = {$P_1$,… , $P_k$} : homogenous top-K candidate preferences

BEGIN
1.  $P^{CAND} \leftarrow \emptyset$, $Rank\_Act \leftarrow \emptyset$, $p \leftarrow 0$
2.  $P^{ACT} \leftarrow Resolve\_conflicts(P^{ACT}, Q)$
3.  Compute $Score(P_i)^{CAC}$ for each $P_i$ in $P^{ACT}$
4.  $Rank\_Act \leftarrow$ Rank $P^{ACT}$ according to $Score(P_i)^{CAC}$
5.  $T^{CAC} \leftarrow BuildTreeC(CAC)$
6.  While ($\exists\ M_i = (P_i, cp_i)$ in $Rank\_Act$) And ($p \leq K$) Do
7.      included $\leftarrow$ True
8.      $T_i \leftarrow BuildTreeC(cp_i)$
9.      While ($\exists$ $edge_j$ in $T_i$) And (included) Do
10.         If ($edge_j \notin T^{CAC}$) Then
11.             included $\leftarrow$ False
12.         End if
13.     End while
14.     If (included = True) Then
15.         $P^{CAND} \leftarrow P^{CAND} \cup P_i$
16.         p++
17.     End if
18.  End while
19.  Return ($P^{CAND}$)
END.
```

Fig. 5. Preference Selection Algorithm

The preference selection algorithm is depicted in Fig. 5. The algorithm takes as input a list of active preferences in the form (($pred_i$, θ_i), cp_j), the number K of preferences to select, and the current analysis context. It outputs the top K candidate preferences. If K is omitted, all candidate preferences are selected.

Example 5. Let's consider the query Q_1. Recall that Prof. Brian candidate preferences are:

P_1 = ($pred_1$; 0.9), P_2 = ($pred_2$; 0.6), P_3 = ($pred_3$; 0.5), and P_4 = (Category='IEEE' OR Category='ACM', 1). P_1 is associated to cp_1, while P_2 and P_3 are related to cp_2, and P_4 is a global preference. The covering rate of cp_1 and cp_2, computed according to (2), are respectively 0.125 and 0.5. The scores of the preferences under *CAC* are respectively: 0.112, 0.3, 0.25, and 1. For K=2, selected preferences are P_2 and P_4. Note that, initially, the interest degree of P_1 (0.9) is greater than the degree of P_2 (0.6).

3.3 Query Enhancement

OLAP offers a rich set of multidimensional operations. However, as the majority of OLAP systems store data in R-OLAP context using fact and dimension tables, all user operations are executed according to SELECT-GROUP-BY-HAVING queries, eventually extended with CUBE and ROLLUP operators [6] (see Fig. 6).

The purpose of the query enhancement step is to integrate the K selected preferences into the user query and produce a new, personalized query that will generate satisfactory results for the user.

```
SELECT D1.a1, ..., Dn.an, [fAGREG ( ] Fi.mj [ )],...
FROM Fi, D1, ...,Dn
WHERE Fi. key1 = D1.keyD1 AND ... AND Fi. keyn = Dn. keyDn
[AND Dj.aj Op Cj, …] [AND Fi.mk Op Ck, …]
GROUP BY [ROLLUP | CUBE] D1.a1, ..., Dn.an,…
[HAVING fAGREG (Fi.mx) Op Cx,…]
[ORDER BY Dy. ay, ...];
```

Fig. 6. OLAP Query Prototype

If there are no homogenous candidate preferences (P^{CAND} is empty), we say that the MDB content is *fully adapted* with the user preferences under the current analysis context. This occurs when all user preferences have been explicitly defined in the query *Q*, or if the user does not have non-conflicting preferences under *CAC*. Therefore, the system executes the query without preprocessing. Otherwise, preferences are included in Q. To this aim, predicates of the selected preferences are embedded in conjunction with those of *Q* within WHERE and HAVING clauses. Furthermore, any additional query element required for the preference predicates are also incorporated into *Q*, *i.e.*, extending FROM clause with new dimension tables, and adding join conditions to WHERE clause.

Predicates on dimension parameters (*e.g.*, year = 2010) and predicates on the non-aggregated measures (*e.g.*, nb_publis > 2) are inserted into the WHERE clause to restrict tuples that will be aggregated. However, predicates on aggregated measures (*e.g.*, SUM (nb_publis) > 4) are inserted into the HAVING clause to eliminate the result cells that do not match with user preferences.

4 Experimental Results

In order to evaluate our framework, we have implemented a java prototype system on top of Oracle 11g. In this section, we present results of our experiments. Data primarily comes from the publication database of our research laboratory (see Fig. 1). Constellation data are stored in R-OLAP context. User profiles (preferences and contexts) are stored in metatables. The MDB consists of two facts, 3 dimensions, 5 measures and 22 attributes. It contains 2 500 000 fact tuples and 500 dimension tuples.

Size of Personalized Content

In the first experiment we measured the change in the size of the MDB obtained from the extension of the original one with user profiles. Recall that a preference may be associated with several contexts, and vice-versa. We chose real user profiles with different number of preferences and various numbers of contexts by preference (*nb. cps* = 0, 2, and 5). For each *nb. cps*, we calculated the percentage of the profiles' size to the size of the initial MDB. Fig. 7 shows this percentage for intervals of 200 preferences. We see that the size of global preferences (*i.e.*, with *nb. cps* = 0) is very small; it does not exceed 0.4% of the size of the original MDB. This is explained by their low storage requirements as they include only atomic predicates (string) and scores between 0 and 1 (float). Besides, for the same preference number range, the profile size is larger when user preferences are associated to more contexts. The difference of size arises from the storage of more contexts and their mappings to preferences. Furthermore, Fig. 7 shows a rapid increase in the profile size for preferences number between 0 and 600. However, we observe a slight increase from 800 preferences. In fact, the total number of contexts, whose size is the major part of the profile size, does not rise significantly from this point. Actually, users do not visit all possible analysis

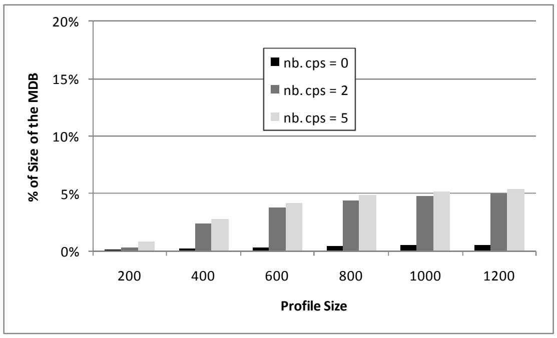


Fig. 7. Impacts of User Profile Storage

contexts, since, in practice, they navigate through a subset of the MDB constellation [3]. So, new preferences are associated with existing contexts (only mappings are stored). However, the whole MDB size may rise significantly for huge profiles. Defining a threshold for the interest degrees of the stored preferences may help at monitoring the profiles' sizes.

Overall, this experiment has shown that the benefits of dynamically personalizing OLAP data can be significant in terms of the storage size (5% from the MDB). The generation of different versions of the MDB at design time for different user types leads to store a huge volume of data, while our framework stores only user profiles.

Efficiency of Preference Selection Algorithm

As the main step of preference selection is the context trees matching, we measured the execution time of our context matching algorithm for 20 synthetic user profiles varying the size of the current analysis context (see Fig. 8 (a)). Such size denotes the number of the context tree edges. User profiles were automatically produced with the use of a profile generator. Then, we measured the algorithm time for different sizes of preference contexts (see Fig. 8 (b)). This size refers to the average size of all context trees in the user profiles.

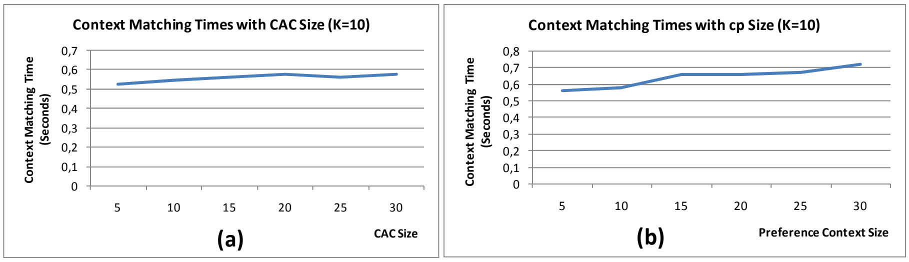


Fig. 8. Efficiency of our Context Resolution Algorithm

As expected the performance of the matching algorithm degrades as the size of context tree increases. However, the execution time does not rise dramatically for large sizes of *CAC* and *cp*. This surprising behavior can be explained considering the way the algorithm proceeds. The algorithm performs a first traversal of the user profile and extracts active preferences with their first and second-level context trees edges. Then, several preference contexts are discarded by a partial context matching without requiring database accesses. The remaining contexts are fully matched with *CAC*. As the selected active preferences are already ranked by their scores under *CAC*, the algorithm stops when the preference context, which matches with *CAC*, of rank K is derived. Therefore, the more the best-ranked active preferences match with *CAC*, the less the required matching iterations are.

Performance of Query Personalization

This experiment aims at evaluating the performance of content personalization process. We ran 10 queries with different values of K over a set of 20 random user profiles. In each run, we calculated the average response time of the personalized queries and the response time of the initial one. As Fig. 9 shows, the overall time for execution of the personalized query is slightly greater than the time required for the execution of the initial one. This is explained by the fact that, as more preferences are integrated (in the form of restriction predicates) into the query, the personalized query returns less rows than the initial one. The decrease in the size of the query results leads to relatively regularize the overall query response time that would be raised with the time for personalization. The results follow the trends expected by the nature of K that allows monitoring the personalized query execution time w.r.t. specific constraints: personalization performs well with K. It is important to notice that, for K = 1, personalized query time is less than the initial query time: the decrease in the content selection time due to data restriction is more important than the response time increase due to selecting one preference.

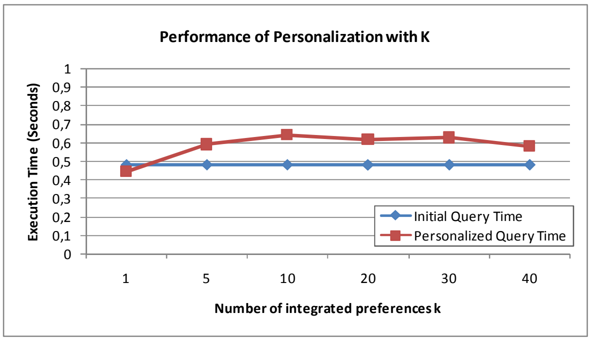


Fig. 9. Performance of Query Personalization w.r.t. K

5 Related Works

A lot of research has been carried out on database personalization [11,12]. For instance, the framework in [12] provides preferences as atomic predicates to personalize the user query. Preferences are independent from user contexts; hence each preference is applicable for every query. This leads to inaccurate results in an OLAP environment, *e.g.*, a preference "year = 2010" which is intuitively related to the analysis of the missions' fees will be also applied when analyzing publications. We enhance preferences with analysis contexts and focus on how context-aware preferences have impact on the multidimensional data selection. Recently, it has been recognized that an ad hoc approach must be devised for dealing with OLAP preferences [15]. We will discuss related works in the OLAP field using the following axes: personalization approaches, and user preference models.

OLAP personalization approaches can be classified into three main categories. *(a) Schema personalization approaches.* The works in [2,7] deal with updating the MDB dimensions and hierarchies in order to adapt the schema to changing requirements. [3] proposes to personalize the OLAP schema at the conceptual level. User models are defined at design time, and then are combined with rules in order to generate user-specific schemas at runtime. In this paper, we propose to personalize dynamically the OLAP content.

(b) Queries personalization approaches. Two approaches have been proposed for personalizing OLAP queries: 1) ranking of query results based either on clustering techniques [18], or on user preferences [5], and 2) query enhancement with dimensions attributes using user-defined rules [14] or using preferences that are defined on the schema structures [8]. These approaches allow personalizing the OLAP structures querying, whereas we deal with the personalization of the content selection. Besides, Bellatreche et *al.* [1] propose a preference-based framework for personalizing user queries visualization w.r.t. specific visualization constraints. Thus, the personalization process is performed after the content selection, while in our framework personalization is performed before.
(c) Queries recommendation approaches aim at suggesting queries to the user in order to assist her in exploring the MDB [4,9]. For a user query *Q*, the system generates in addition to the query result, further queries $Q_1, \ldots Q_n$ that have been posed by the users [4], or that are computed incrementally based on user preferences [9]. Our approach, however, consists in integrating preferences into the query Q, then generating an enhanced query *Q'*, such that *Q* is syntactically included in *Q'*. Our content personalization approach may be applied at the end of a recommendation process to more customize the recommended queries.

User preferences are expressed as strict partial order (*e.g,*. I prefer A to B) [1,5] or may be associated with a number indicating the user interest degree [10]. [1,5] cannot capture different degrees of interest, such as 'I like journal papers very much', 'I like publications about information retrieval a little'. Besides, user preferences are expressed as (a) hard constraints, that are either satisfied or not satisfied at all [3,8]; or (b) soft constraints, that should be fulfilled as closely as possible [5]. In this paper, we are concerned with preferences expressed as hard constraints. Each preference is associated to a number, which, in our case, indicates the user interest in results that exactly satisfy this preference. Incorporating other types of preferences within our framework is part of ongoing work. In [5], the user preferences are integrated within the query using preference constructors. The focus of our work is different, since we are interested by preferences as long-term information needs stored in a user profile, and then automatically extracted at query time saving user effort. Finally, our model differs from the above models since preferences are context-aware. Their selection is driven by a context resolution process.

6 Conclusions

Providing customized content is crucial to better satisfying users. One solution here is static personalization, where different versions of the MDB are generated at design time for different user types. Such solution will result in both storing a huge volume of data and implanting several ETL processes. In this paper, we propose a framework for dynamic personalization where the MDB content is built at runtime depending on the user preferences and analysis context. User preferences are defined as predicates on the OLAP content, and are associated with analysis contexts of different levels of detail. At runtime, content selection is based on a combination of the user query and preferences stored in a profile in order to provide a personalized content. We presented personalization algorithms and experimental results showing the efficiency of our algorithms.

Currently, we are focusing on a mining technique that acquires user preferences by inferring their data analysis behavior. Our future work will focus on the following issues: we intend to extend our preference model in order to support more types of preferences, such negative and soft ones, and to personalize content selection using these types; and we plan to evaluate various scoring functions for preferences.

References

1. Bellatreche, L., Giacometti, A., Marcel, P., Mouloudi, H., Laurent, D.: A personalization framework for OLAP queries. In: DOLAP, pp. 9–18. ACM, New York (2005)
2. Favre, C., Bentayeb, F., Boussaid, O.: Evolution of Data Warehouses' Optimization: a Workload Perspective. In: Song, I.-Y., Eder, J., Nguyen, T.M. (eds.) DaWaK 2007. LNCS, vol. 4654, pp. 13–22. Springer, Heidelberg (2007)
3. Garrigós, I., Pardillo, J., Mazón, J., Trujillo, J.: A Conceptual Modeling Approach for OLAP Personalization. In: Laender, A.H.F., et al. (eds.) ER 2009. LNCS, vol. 5829, pp. 401–414. Springer, Heidelberg (2009)
4. Giacometti, A., Marcel, P., Negre, E.: Recommending Multidimensional Queries. In: Pedersen, T.B., Mohania, M.K., Tjoa, A.M. (eds.) DaWaK 2009. LNCS, vol. 5691, pp. 453–466. Springer, Heidelberg (2009)
5. Golfarelli, M., Rizzi, S.: Expressing OLAP Preferences. In: Winslett, M. (ed.) SSDBM 2009. LNCS, vol. 5566, pp. 83–91. Springer, Heidelberg (2009)
6. Gray, J., Bosworth, A., Layman, A., Pirahesh, H.: Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-total. In: ICDE, pp. 152–159 (1996)
7. Hurtado, C.A., Mendelzon, A.O., Vaisman, A.A.: Maintaining Data Cubes under Dimension Updates. In: ICDE, pp. 346–355. IEEE Computer Society, Los Alamitos (1999)
8. Jerbi, H., Ravat, F., Teste, O., Zurfluh, G.: Management of Context-aware Preferences in Multidimensional Databases. In: IEEE ICDIM, pp. 669–675 (2008)
9. Jerbi, H., Ravat, F., Teste, O., Zurfluh, G.: Applying Recommendation Technology in OLAP Systems. In: Filipe, J., Cordeiro, J. (eds.) ICEIS 2009. LNBIP, vol. 24, pp. 220–233. Springer, Heidelberg (2009)
10. Jerbi, H., Ravat, F., Teste, O., Zurfluh, G.: Preference-Based Recommendations for OLAP Analysis. In: Pedersen, T.B., Mohania, M.K., Tjoa, A.M. (eds.) DaWaK 2009. LNCS, vol. 5691, pp. 467–478. Springer, Heidelberg (2009)
11. Kießling, W.: Foundations of preferences in database systems. In: VLDB, pp. 311–322 (2002)
12. Koutrika, G., Ioannidis, Y.E.: Personalized Queries under a Generalized Preference Model. In: International Conference on Data Engineering, pp. 841–852 (2005)
13. Ravat, F., Teste, O., Tournier, R., Zurfluh, G.: Algebraic and graphic languages for OLAP manipulations. International Journal of Data Warehousing and Mining 4(1), 17–46 (2008)
14. Ravat, F., Teste, O.: Personalization and OLAP databases. In: Volume New Trends in Data Warehousing and Data Analysis of Annals of Information Systems, pp. 71–92 (2009)
15. Rizzi, S.: OLAP preferences: a research agenda. In: DOLAP, pp. 99–100 (2007)
16. Simitsis, A., Vassiliadis, P., Sellis, T.: State-Space Optimization of ETL Workflows. IEEE Transactions on Knowledge and Data Engineering 17(10), 1404–1419 (2005)
17. Smyth, B., Bradley, K., Rafter, R.: Personalization Techniques for Online Recruitment Services. Communications of the ACM 45(5), 39–40 (2002)
18. Xin, D., Han, J., Cheng, H., Li, X.: Answering top-k queries with multi-dimensional selections: The ranking cube approach. In: VLDB, pp. 463–475 (2006)

A Coverage Representation Model Based on Explicit Topology

Salahaldin Juba and Peter Baumann

Jacobs University, 28759 Bremen, Germany

Abstract. In geoservices, coverages denote space-time varying extended phenomena. Examples include sensor observation time series, raster images (like hyperspectral satellite imagery), and climate and ocean data. Coverage properties include both topological (e.g., neighborhood) and geometrical (e.g., resolution) elements. Obviously, it is paramount for open geoservices to have a well-defined coverage data model which can lead to a more efficient processing of spatial queries.

The representation model proposed here is based on the cw-complex notion and adds a geometrical realization to each topological primitive in the cw-complex. This model can capture structured coverages such as raster grids and unstructured coverages such as triangulated irregular networks (TINs). Also, this model provides a conceptual model for manipulating coverages in scientific database systems.

Keywords: Coverages, spatial databases, topology.

1 Introduction

Coverage and field terms can be used interchangeably in the literature to express space-time varying phenomena. coverage/field can be considered a function ($F : D \rightarrow R$) that maps specific location identified by spatial coordinates (domain D) to physical values (range R) such as temperature and radiometric reflectance. Coverage domain and range depend on their application which might be a capturing device such as remote sensor or a simulation model. Coverage dynamics and properties vary from one coverage type to another. In ISO 19123 [7], coverages are described based on there internal cell structure as shown in the Tables 1 and 2.

Coverages classification according to the internal cell structure [7] facilitate the description of specific coverage functions and properties[1]. However, it is very complex to extend this standard to support other coverages because it does not provide a coherent definition for different coverage types. In this model, coverage definition is based on the combinatorial structure of cells constructed from topological primitive elements including vertices, edges, faces, and volumes glued together in an inductive way.

[1] ISO19123 is approved for reversion by ISO/TC 211 programme maintenance group (PMG).

Table 1. Discrete coverage internal cell structure

Discrete coverages	
Coverage Type	Internal cell structure
CV_DiscretePointCoverage	point
CV_DiscreteGridPointCoverage	point
CV_DiscreteCurveCoverage	line segment
CV_DiscreteSurfaceCoverage	polygon
CV_DiscreteSolidCoverage	volume

Table 2. Continuous coverage internal cell structure

Continuous coverages	
Coverage Type	Internal cell structure
CV_ContinuousThiessenPolygonCoverage	point
CV_ContinuousQuadrilateralGridCoverage	point
CV_ContinuousTINCoverage	triangle
CV_ContinuousHexagonalcoverage	hexagon
CV_ContinuousSegmentedCurvecoverage	line

2 Previous Work

Sharing scientific data is required for collaborative cooperation between scientists and commercial/noncommercial services provision. There are already defined services and standards to achieve this goal such as web coverage service (WCS)[12]. However, because the underlying data models where these standards are built on are defined for a specific coverage types, up till now, there is no a coherent framework for serving different coverage types.

Some of the existing specialized data models, application programming interfaces (APIs) and protocols such as raster data array manager (rasdaman)[13], OpenDAP[11], HDF[4], and NetCDF[10] provide the means for storing, manipulating and sharing array-based scientific data such as time series raster images. Other services, such as FieldGML[8], assume that continuous phenomena in space time is acquired by the process of discretization and interpolation. In order to share a coverage, FieldGML ships the sample data and the interpolation methods that are required to build the coverage to the client. FieldGML model can save the network bandwidth but it assumes that clients have sufficient processing resources. In the area of computer-aided design and manufacturing (CAD/CAM), the existing models, such as boundary representation (b-rep) and constructive solid geometry (CSV), do not provide a suitable operation set for manipulating coverages. However, they give a clue how to define the coverage domain. Furthermore, most of the 3D GIS models are utilizing CAD models such as b-rep and CSG or 3D TEN (Tetrahedral network)[1]. In general, these models are suited for discrete object representations, and can not represent multi-dimensional data.

The coverage representation model is highly influenced by gridfield model [6]. The gridfield model defines a set of minimal closed operations for both structured and unstructured grids. One drawback of gridfield model is that it treats the geometrical data exactly as fields and totally based on topological attributes of grids. Also, gridfield model is very general and grid constraints need to be validated.

3 Coverage Model

Topology concerned with spatial properties that are preserved under continuous deformations of objects. Encoding topological properties explicitly is very useful in answering certain geographical queries such as adjacency, and connectivity. Topological spaces are easier to manipulate and process in computers than geometrical spaces since computers can not be used to completely represent continuity based on Euclidean distance [3]. For example, in processing geometrical object, the rounding function may lead to wrong spatial relationship such as containment. Algorithms that are immune to floating-point errors are introduced by [9]. Currently, many of the n-dimension models and applications are based on topology such 3D city, simplified spatial model(SSM), urban spatial model (USM), etc[15].

Coverage domain can be represented as a cell complex X. Cell complex is any topological space that can be constructed form the union of non-intersecting cells. Cells can be restricted to very special cases such as triangles (simplicial complex) or can allow individual cells to be attached to one another in generic way (cw-complex). Mathematically, cell complex is based on the definition of open cells

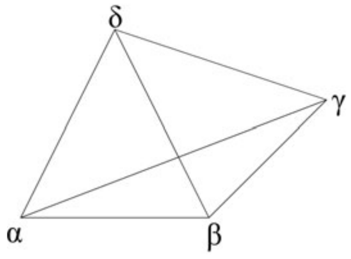
Definition 1. *Let X be a Hausdorff space. A set $c \subset X$ is an open $k-cell$ if it is homeomorphic to the interior of the open k-dimensional ball $D^k = \{x \in R^k | \parallel x \parallel < 1\}$. The number k is unique by the invariance of domain theorem, and is called dimension of c.*

Definition 2. *A cell complex constructed inductively indicated as $X_k = \{D_i, 0 \leq i \leq k\}$ Where D_i is called space partition or subspace, i is the subspace dimension, k is the cell complex dimension, and $i, k \in N$.*

Definition 3. *The space partition D_i is a set of all cells that have the same topological dimension i.*

The above definitions are general enough to represent different coverage types. For example, the coverage domain of xyz $3D$ linear meshes can be represented as a cell complex X_3 such that:

$D_0 \subseteq \{a_0, a_2, ..., a_n\}$ where a_i is a 0-cell, $i, n \in N$.
$D_1 \subseteq \{a \in \rho(D_0) | \, |a| = 2\}$
$D_2 \subseteq \{a \in \rho(D_1) | \, |a| \geq 3\}$
$D_3 \subseteq \{a \in \rho(D_2) | \, |a| \geq 4\}$



D0 = {α, β, γ, δ}
D1 = {{α, β}, {α, γ}, {α, δ}, {β, γ}, {β, δ}, {γ, δ}}
D2 = {{{α, β}, {α, γ}, {β, γ}},
{{α, β}, {α, δ}, {β, δ}},
{{α, γ}, {α, δ}, {γ, δ}},
{{β, γ}, {β, δ}, {γ, δ}}}
D3 = {{{{α, β}, {α, γ}, {β, γ}},
{{α, β}, {α, δ}, {β, δ}},
{{α, γ}, {α, δ}, {γ, δ}},
{{β, γ}, {β, δ}, {γ, δ}}}}

Fig. 1. Cuboid topology

Where $\rho()$ indicates power set, a indicates n-cell, and $||$ indicates cardinality constraints

The space partitions D_0, D_1, D_2, D_3 represents a set of vertices, edges, faces, and volumes respectively. The cardinality constraints define the number of minimum cells required to construct a higher cells. For example, the minimum number of 1-cells needed to construct a 2-cell is three. The coverage domain of xyz $3D$ can be altered to support time, such that $D_4 = \{a \in \rho(D_3)| \ |a| = 2\}$ or to support other coverage types such as time series point observation. The coverage definition is based on incident relationship[2,6,5]. For example, $a \preceq b$ reads as a indecent to b if $a \in b$.

This combinatorial structure of cell complex allows data binding for different cell complex partition [6]. In this context, the coverage can be defined as a set of functions that map each space partition D_i to a certain range R_i, that is $C = \{f_i, ..., f_k\}$ where $f_i : D_i \mapsto R_i$ for $0 \leq i \leq k$.

3.1 Coverage Topological Constraints

The combinatorial structure of cell complex may lead to the construction of improper meshes. To insure the generation of well-structured cell complexes certain rules can be validated. Note that, the coverage domain topological constraints are not necessarily mutually exclusive.

1. Coverage cells do not coincide, i.e., they are not redundant, that is, $(\forall a, b \in X_k | a \neq b)$
2. Coverage domain should be homogeneous, that is, $(\forall a \in X_k | k > 0)$ there exists $b \in X_k$ such that $a \in b$. Homogeneity ensures that nD grid is a set on $n - cells$ where isolated vertices and dangling edges are not allowed.
3. The coverage domain is strongly connected, that is, given X_k, any two k-cells can be joined by a "chain" of k-cells in which each pair of neighboring cells have a common $k - 1 - cell$.
4. The coverage domain do not branch, i.e., given X_k coverage domain each $K - 1 - cell$ is indecent to precisely two k-Cells.

Fig 2. depicts different invalid coverage domains. Homogeneity, branching and connectivity are violated by Fig 2. a, b and c respectively.

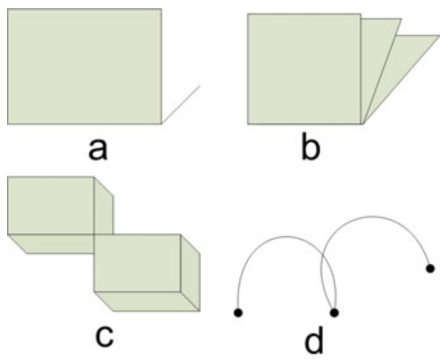


Fig. 2. Invalid coverage domain

The validity of certain coverage domain constraints such connectivity can be achieved by the employment of Euler-Poincarè Formula. The Euler-Poincarè formula of space X_k can be given by:

$$\chi(X_k) = \sum_{i=0}^{k} (-1)^i |a_i| = k \tag{1}$$

Where $|a_i|$ represents the number of cells which have the i_{th} topological dimension. For $2D$ spaces, $\chi(X_2) = 2$, which reduce to classical Euler formula

$$V - E + F = 2 \tag{2}$$

Cardinality constraints can be altered to capture different coverage types. For example, xyz $3D$ domain can be altered to capture only TEN meshes by changing the cardinality constraints of D_2 and D_3 subspaces as follows:

1. $D_2 \subseteq \{a \in \rho(D_1) | \ |a| = 3\}$
2. $D_3 \subseteq \{a \in \rho(D_2) | \ |a| = 4\}$

3.2 The Geometric Realization of the Coverage Model

Topology and geometry have a tight relationship and in some cases they are dependent. For example, point set topology was developed using metric (distance) relationships. Furthermore, grid dimensionality is an attribute that depends on both topological and geometrical attributes. To illustrate, one can not construct 4-dimensional object in $2D$ space.

The definition of cell complex do not take into account geometrical properties. This issue can be treated by mapping each element in the coverage domain to its geometrical realization. For example, given a cell complex X_k which represent a linear coverage domain such as the Cartesian grid. The geometrical realization of the coverage domain can be achieved by mapping D_0 to a set of vectors in k-dimensional euclidean space such that $a_i \mapsto \hat{V}$ where $\hat{V} \in R^k, a_i \in D_0, i \in N$.

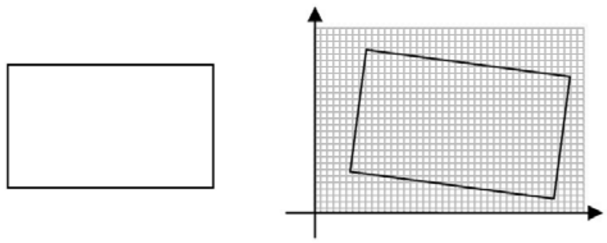


Fig. 3. On the left: Topological space, on the right: its geometrical realization

3.3 Geometrical Constraints of Coverages

The addition of the geometrical realization requires more data integrity checks.

1. 0-cells do not coincide, i.e. they have different assigned vectors in the euclidean space. That is $(\forall x, y \in D_0 | x \neq y) \Rightarrow (\hat{X} \neq \hat{Y} | x \mapsto \hat{X}, y \mapsto \hat{Y})$.
2. Any two cells in the same coverage are not geometrically intersected. i.e. $(\forall x, y \in X_k | x \neq y) \Rightarrow (intersect(x, y) = \phi)$. See Fig.2 d.

The first condition ensures that each vertex is unique which in turn can be used to support the topological integrity constraints. The second condition, ensures that each point in space has a valid data assigned to it.

4 Coverage Model Benefits

The generality of the coverage representation model can lead to the provision of a coherent models for representing spatial data as follows:

- Coherent model for scientific data: The coverage model can represent different scientific data sets including structured/ unstructured meshes. Different scientific communities utilize different coverage types. For example, in ecology hexagonal girds are widely used to represent movement paths. Rectilinear grids are used to represent different spatial resolutions and it is used in near ocean simulations.
- Coherent model for raster data interpretation: In GIS, raster images can be interpreted as point raster and as cell raster. In point raster, each point represents an observation in the euclidean space and it has an exact location. In cell raster, each cell represents an area segment. Digital elevation model(DEM) falls in point raster classification and land use raster images falls in cell raster interpretation.

  The cell complex definition $X_k = \{D_i, 0 \leq i \leq k\}$ can capture both cases. Point raster, is constructed from D_0 subspace while cell raster is constructed from D_0, D_1, and D_2 subspaces. Furthermore, raster data model is implemented as a multidimensional arrays where the array index either represents the top left cell corner or cell center. These two cell coordinate systems, center-based and upperleft-based, may lead to overlaying problems in raster algebra. In this model, these errors can be found immediately by comparing the geometrical realization of the 0-cells of the raster grid.

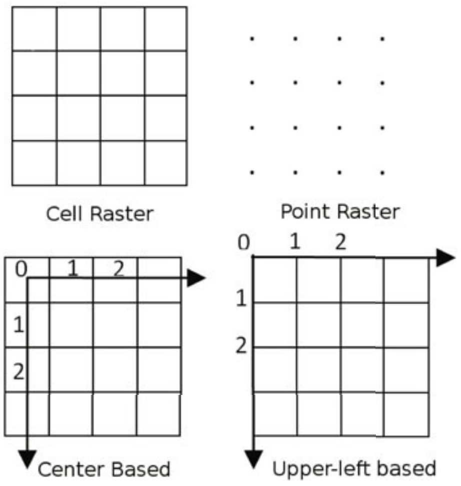


Fig. 4. Different interpretation of raster images

5 Coverage Model Operations

Operators are defined on three levels, which are cell, space partition and coverage and can be classified to geometrical and topological operators. The *Bind* operation [6] (simply the constructor) assigns metadata information to coverages, and binds subspaces to physical values. In GIS, metadata is very important not only for social and reference purposes but also for processing purposes. For example, even if the coverage cells have geometrical realization(e.g. coordinates for 0-cells), it is impossible to project coverages if they are not assigned spatial reference information.

The assignment of metric data to each n-cell allows the calculation of grid metric attributes such as distance between two cells, length of line segment, polygon perimeter, cell area, cell volume, the total coverage area and the relation between cells such as containment, and intersection. For example, The Hausdorff distance of two cells A and B in the cell complex X_k can be calculated by the following algorithm. For every 0-cell a of A, find its smallest distance to any 0-cell b of B and keep the smallest distance found among all points. That is $D(A,B) = min_{a \preceq A}\{min_{b \preceq B}\{d(a,b)\}\}$ where $d(a,b) = (\sum_{i=1}^{k} |x_i + y_i|^2)^{\frac{1}{2}}$ where $a, b \in D_0, a \mapsto \hat{x}, b \mapsto \hat{y}$ and k is the length of x and y vectors and i is the vector index. The calculation of geometrical attributes such as distance and volumes returns a scalar values which can be used as predicates in spatial queries such as select all 2-cells from a coverage where the area is greater that a specific value.

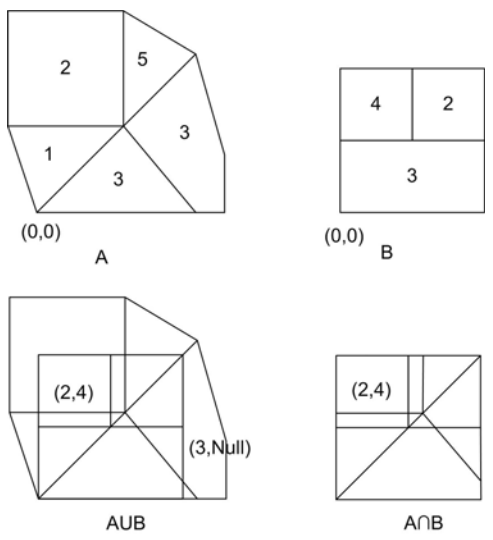


Fig. 5. The intersection and the union of two coverages

A more complicated operations are the overlaying operations such as intersect and union of different coverages which in turn creates a new coverage. Fig. 5 shows an example of two coverages intersection and union based on geometrical attributes. This kind of operation is quite complex, the attributes from both coverages are merged leading to null values. Furthermore, the new coverage domain topology needs to be reconstructed properly.

Regarding the topological operations, gridfield algebra defines a minimal algebraic operations for handling grids. These operators can handle both coverage domain and attributes. The following gives a brief description of these operators:

- Restrict operator: works as a relational select, the restrict operator ensure that the resulting gridfield domain is well supported after applying the predicate.
- Accrete operator: grows a gridfield by attaching cells from another gridfield.
- Merge operator: find the topological intersection between two gridfields and retains the data values defined over this intersection.
- Cross Product: create a higher-dimensional gridfield from two lower-dimensional gridfields.
- Regrid: maps a source gridfield cell values to target gridfield cells and then aggregates the bounded data and binds it to the target coverage.

6 Case Study

The current WCS standard offers three operations which are getCapabilities, describeCoverage and getCoverage. The first two operations gives a brief description and detailed description a bout the server offering and can be served

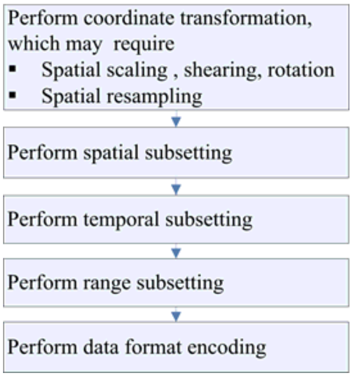

Perform coordinate transformation, which may require
▪ Spatial scaling , shearing, rotation
▪ Spatial resampling
Perform spatial subsetting
Perform temporal subsetting
Perform range subsetting
Perform data format encoding

Fig. 6. Conceptual GetCoverage evaluation sequence, src:[12]

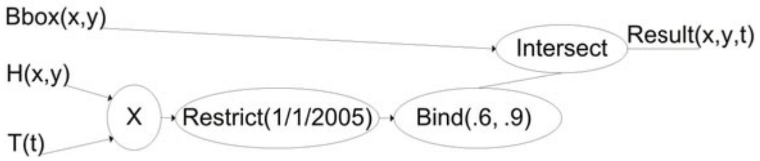


Fig. 7. GetCoverage evaluation scenario. Topological and geometrical operators are represented in different colors

by querying the assigned coverage metadata. The most complicated operation is getCoverage which might require different processing stages. Each getCoverage processing step can be mapped to a certain coverage representation model operation. The spatial subsetting can be achieved by employment of the geometrical intersection. Also, the Bind operation can be employed to perform range subsetting. The coverage model operations are suited for processing getCoverage operation since GetCoverage operations preserve the coverage underlying topology.

The following scenario shows how WCS getCoverage operation can be evaluated. Assume a coverage (time series raster images) with radiance. Radiance has two axes, wave length with three keys(0.6, 0.7, 0.9) and forecast time with four keys (1/1/2005, 1/1/2006, 1/1/2007, 1/1/2008). A simple getCoverage request can be formulated as the following: select 0.6 and 0.7 radiance at 1/1/2005 within a certain bounding box (BBox). This query can be answered as shown in Fig. 7. First, the coverage domain can be constructed by applying the cross product and restrict operation. After that, the coverage domain can be mapped to the suitable wave lengths and intersected with BBox.

7 Discussion and Future Work

The coverage representation model can be used to process manipulate and share scientific data and one single set of operations is available for regular meshes and irregular meshes. Also, this model can handle some of the raster data model drawbacks such as closed curve theorem and Egenhofer relations[14].

This model tackles gridfield limitations and adds extra constraints and functionalities based on the geometrical properties of grids to capture coverages properly. Coverage spatial queries requires the combination of different operations which in turn leads to the generation of different execution plans. For example, in the WCS scenario the bind operation and the restrict operation can be swapped. This model provides the basis for defining a query language for manipulating coverages.

Currently we are working on additionally modeling special coverage types such as curvilinear grids.

Acknowledgement

This Work is supported by ESA under contract HMA-FO Task 3.

References

1. Abdul-Rahman, A., Pilouk, M. (eds.): Spatial Data Modelling for 3D GIS. Springer, Heidelberg (2008)
2. Berti, G.: Generic Software Components for Scientic Computing. PhD thesis, Technical University Cottbus (2000)
3. Franklin, W.: Cartographic errors symptomatic of underlying algebra problems, pp. 190–208 (1984)
4. Hartnett, E.: Merging the NetCDF and HDF5 Libraries to Achieve Gains in Performance. In: Earth Science Technology Conference, ESTC (2004)
5. Heinzl, R.: Data structure properties for scientific computing: an algebraic topology library. In: POOSC 2009: Proceedings of the 8th Workshop on Parallel/High-Performance Object-Oriented Scientific Computing, pp. 1–6. ACM, New York (2009)
6. Howe, B.: Gridfields: Model-Driven Data Transformation in the Physical Sciences. PhD thesis, Portland State University (2007)
7. ISO. The OpenGIS Abstract Specification Topic 6, Schema for Coverage Geometry and Functions (August 2003)
8. Ledoux, H.: Representing Continuous Geographical Phenomena with FieldGML. Technical report, Geo-Database Management Center (2008)
9. Mount, D.M.: Geometric intersection. In: Handbook of Discrete and Computational Geometry, pp. 615–630. CRC Press, Inc., Boca Raton (1997)
10. Rew, R., Davis, G.: NetCDF: An Interface for Scientific Data Access. IEEE Computer Graphics and Applications 10, 76–82 (1990)
11. Sgouros, T.: OPeNDAP User Guide (July 2007)
12. Whiteside, A., Evans, J.: Web Coverage Service (WCS) Implementation Standard (March 2008)

13. Widmann, N., Baumann, P.: Efficient execution of operations in a dbms for multidimensional arrays. In: SSDBM 1998: Proceedings of the 10th International Conference on Scientific and Statistical Database Management, pp. 155–165 (July 1998)
14. Winter, S., Frank, A.: Topology in raster and vector representation. GeoInformatica 4, 35–65 (2004)
15. Yanbig, W., Lixin, W., Wenzhing, S., Xiaomeng, L. (eds.): On 3D GIS Spatial Modeling, ISPRS Workshop on Updating Geo-spatial Databases with Imagery & The 5th ISPRS Workshop on DMGISs. ISPRS (2007)

Exact and Efficient Proximity Graph Computation

Michail Kazimianec and Nikolaus Augsten

Faculty of Computer Science, Free University of Bozen-Bolzano,
Dominikanerplatz 3, 39100 Bozen, Italy
`kazimianec@inf.unibz.it, augsten@inf.unibz.it`

Abstract. Graph Proximity Cleansing (GPC) is a string clustering algorithm that automatically detects cluster borders and has been successfully used for string cleansing. For each potential cluster a so-called proximity graph is computed, and the cluster border is detected based on the proximity graph. Unfortunately, the computation of the proximity graph is expensive and the state-of-the-art GPC algorithms only approximate the proximity graph using a sampling technique.

In this paper we propose two efficient algorithms for the exact computation of proximity graphs. The first algorithm, PG-DS, is based on a divide-skip technique for merging inverted lists, the second algorithm, PG-SM, uses a sort-merge join strategy to compute the proximity graph. While the state-of-the-art solutions only approximate the correct proximity graph, our algorithms are exact. We experimentally evaluate our solution on large real world datasets and show that our algorithms are faster than the sampling-based approximation algorithms, even for very small sample sizes.

1 Introduction

String data is omnipresent and appears in a wide range of applications. Often string data must be partitioned into clusters of similar strings, for example, for cleansing noisy data. Cleansing approaches that are based on clustering substitute all strings in a cluster by the most frequent string. Such cleansing methods are typically applied for non-dictionary strings since for such data the cleansing algorithm can not take advantage of a reference table with the correct values. Non-dictionary data are, for example, personal data like name and address, or proper names in geography, biology, or meteorology (e.g., names of geographic regions, plants, cyclones, and hurricanes).

Recently, Mazeika and Böhlen [1] introduced the graph proximity cleansing (GPC) method for clustering and cleansing non-dictionary strings. A distinguishing feature of the GPC clustering is the automatic detection of the cluster borders using a so-called proximity graph. The proximity graph measures the number of strings within different neighborhoods of the cluster center.

The computation of the proximity graphs is the bottleneck in GPC algorithms. The proximity graph is expensive to compute and must be computed for each potential cluster. To make GPC feasible for realistic datasets, Mazeika and Böhlen [1] propose an algorithm that *approximates* the proximity graphs using a sampling technique on an inverted list index. The approximated proximity graphs are then used to decide the cluster borders. But the approximate proximity graphs are different from the exact ones and thus lead to errors in the clusters.

In this paper we present two efficient GPC algorithms that compute the *exact* proximity graph. The first algorithm, PG-SM, is based on a sort-merge technique; the second algorithm, PG-DS, uses an inverted list index and a divide-skip strategy to compute the proximity graph. We experimentally evaluate our exact algorithms on large real-world datasets (for example, misspellings of the Oxford text archives) and show that our algorithms are faster than the previously proposed sampling algorithm even for small samples.

Summarizing, our contribution is the following:

– We propose two new algorithms, PG-SM and PG-DS, for the exact computation of GPC clusters. To the best of our knowledge, this is the first work to propose an *exact* solution for GPC.
– We experimentally evaluate our solution on large real-world datasets and show the scalability of our approach.
– We compare to the state-of-the-art GPC algorithm that is based on sampling and only approximates the GPC clusters. We show that our exact algorithm is faster even for small samples, for which the approximation error is large.

The remaining paper is organized as follows. In Section 2 we introduce basic concepts and the state-of-the-art GPC algorithm. We define the problem in Section 3. In Section 4 we present our solution for GPC, in particular, a novel algorithm for computing the cluster center and two efficient algorithms, PG-SM and PG-DS, for computing the exact proximity graph. We experimentally evaluate our solution on large real world datasets in Section 5 and discuss related work in Section 6. In Section 7 we draw conclusions and point to future work.

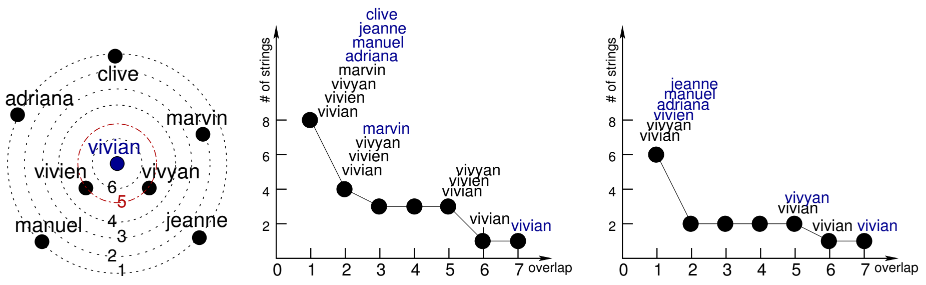
2 Background

In this section we introduce the proximity graph, the GPC method, and the state-of-the-art GPC algorithm.

2.1 Proximity Graph

Given a string s, the **extended string** $\bar{s}$ is s prefixed and suffixed with $q-1$ dummy characters '#'. A ***q*-gram**, κ, of s is a substring of length q of the extended string $\bar{s}$. The **profile** $P(s,q)$ of s is the set of all pairs (κ, i), where κ is a q-gram of s, and i is the i-th occurrence of κ in $\bar{s}$. We use the term q-gram interchangeably for κ and the pair (κ, i), and denote (κ, i) with κ^i.

The **overlap** $o(P(s,q), P(u,q)) = |P(s,q) \cap P(u,q)|$ between two profiles $P(s,q)$ and $P(u,q)$ is defined as the number of q-grams that two profiles share. Intuitively, two strings are similar if they have a large overlap.



(a) Neighborhoods (b) Exact Proximity Graph (c) Approx. Proximity Graph

Fig. 1. Neighborhoods, Exact and Approximate Proximity Graphs for 'vivian'

The **dataset**, D, is a set of strings. The **neighborhood** of profile P in dataset D for overlap threshold τ is the set of all strings of D which have an overlap of at least τ with P, $N(D, P, \tau) = \{s \in D : o(P, P(s, q)) \geq \tau\}$. If P and D are clear from the context, we write N_τ for $N(D, P, \tau)$.

Example 1. The profiles of the strings $s = vivian$ and $u = vivien$ for $q = 2$[1] are $P(s, q) = \{\#v^1, vi^1, iv^1, vi^2, ia^1, an^1, n\#^1\}$ and $P(u, q) = \{\#v^1, vi^1, iv^1, vi^2, ie^1, en^1, n\#^1\}$. The overlap between the profiles of s and u is $o(P(s, q), P(u, q)) = |\{\#v^1, vi^1, iv^1, vi^2, n\#^1\}| = 5$. The neighborhood of the profile $P_1 = P(s_1, q)$ in the dataset $D = \{s_1, s_2, \ldots, s_8\} = \{vivian, adriana, vivien, marvin, vivyan, manuel, jeanne, clive\}$ for overlap threshold $\tau = 3$ is $N(D, P_1, \tau) = \{s_1, s_3, s_5\}$. Figure 1(a) illustrates the neighborhoods of P_1 in D.

The **center** $P_c(N_\tau, q)$ of the neighborhood $N_\tau = N(D, P, \tau)$ is a profile that consists of the K most frequent q-grams in B, where $B = \biguplus_{s \in N_\tau} P(s, q)$ is the bag of all q-grams of neighborhood N_τ, and $K = \lfloor \frac{|B|}{|N_\tau|} + 0.5 \rfloor$ is the average size of the profiles in N_τ. Formally, a profile P is a center of the neighborhood N_τ iff (i) $P \subset B$, (ii) $|P| = K$, and (iii) $\forall \kappa' \in P \; \forall \kappa \in B \setminus P : \; \phi(\kappa', B) \geq \phi(\kappa, B)$, where $\phi(\kappa, B)$ is the number of occurrences of κ in B.

Example 2. We continue Example 1 and compute the center of the neighborhood $N_3 = N(D, P_1, 3) = \{s_1, s_3, s_5\}$: $B = P(s_1, q) \uplus P(s_3, q) \uplus P(s_5, q) = \{\#v^1, \#v^1, \#v^1, vi^1, vi^1, vi^1, iv^1, iv^1, iv^1, vi^2, vi^2, ia^1, an^1, an^1, n\#^1, n\#^1, n\#^1, ie^1, en^1, vy^1, ya^1\}$, $K = \lfloor \frac{21}{3} + 0.5 \rfloor = 7$, and the center of N_3 is $P_c(N_3, q) = \{\#v^1, vi^1, iv^1, n\#^1, vi^2, an^1, ia^1\}$.

The **proximity graph**, $PG(s, D, q) = ((1, |N_1|), (2, |N_2|), \ldots, (k, |N_k|))$, $k = |P(s, q)|$, of string $s \in D$ maps each overlap thresholds τ, $1 \leq \tau \leq k$, to the sizes of the respective neighborhoods N_τ, where N_τ is recursively defined as follows:

$$N_\tau = \begin{cases} \{s\} & \text{if } \tau = |P(s, q)|, \\ N(D, P_c(N_{\tau+1}, q), \tau) \cup N_{\tau+1} & \text{otherwise} \end{cases} \tag{1}$$

[1] In the following examples we use $q = 2$ by default.

Intuitively, the proximity graph shows the neighborhood sizes of the string s for all possible overlap thresholds. The maximum overlap is the size of the string profile, the minimum overlap is 1 (the trivial case $\tau = 0$, for which the neighborhood is D, is not considered). A large overlap threshold requires all strings in the respective neighborhood to be similar to s. Thus the size of the neighborhood *increases* if the overlap threshold *decreases*.

The proximity graph of a string s is computed from right to left, i.e., from the largest to the smallest overlap. The neighborhood of the rightmost point in the proximity graph is defined to be $\{s\}$. For the other points the neighborhood is computed around the center of the previous neighborhood, as defined in (1). The union with $N_{\tau+1}$ guarantees that the proximity graph is a monotonic decreasing function even though the center changes.

Example 3. We continue the previous example and compute the proximity graph for the string $s_1 = vivian$. The overlap threshold ranges between $\tau = 1$ and $\tau = |P(s_1, q)| = 7$. For the maximum overlap threshold the neighborhood is $N_7 = \{s_1\}$ by definition. The neighborhood N_6 is computed around the center $P_c(N_7, q) = \{\#v^1, vi^1, iv^1, vi^2, ia^1, an^1, n\#^1\}$ of N_7, the neighborhood N_5 around the center of N_6, etc. Table 1 shows the neighborhoods N_τ and their centers for all values of τ. The resulting proximity graph $PG(s_1, D, q) = \{(1, 8), (2, 4), (3, 3), (4, 3), (5, 3), (6, 1), (7, 1)\}$ is illustrated in Figure 1(b).

2.2 Proximity Graph Cleansing

We discuss the Graph Proximity Cleansing (GPC) method for clustering strings. GPC takes a string dataset D as input and returns a set of clusters. The clusters are mutually disjoint and cover the dataset D, i.e., no clusters overlap and the union of all clusters is the dataset D.

The pseudo-code for GPC is given in Algorithm 1. GPC randomly picks a string s from the set of eligible cluster centers, which initially is the dataset D. The cluster for s consists of strings in the neighborhood of s, i.e., strings similar to s. The size of the neighborhood that forms the cluster is computed based on the proximity graph of s, as detailed below. The new cluster is added to the set of clusters, and all strings of the new cluster are removed from the set of eligible

Table 1. Computation of the Proximity Graph $PG(vivian, D, q)$ in Example 3

τ	Neighborhood N_τ	Center of N_τ
7	$\{s_1\}$	$\{\#v^1, vi^1, iv^1, vi^2, ia^1, an^1, n\#^1\}$
6	$\{s_1\}$	$\{\#v^1, vi^1, iv^1, vi^2, ia^1, an^1, n\#^1\}$
5	$\{s_1, s_3, s_5\}$	$\{\#v^1, vi^1, iv^1, vi^2, ya^1, an^1, n\#^1\}$
4	$\{s_1, s_3, s_5\}$	$\{\#v^1, vi^1, iv^1, vi^2, ya^1, an^1, n\#^1\}$
3	$\{s_1, s_3, s_5\}$	$\{\#v^1, vi^1, iv^1, vi^2, ya^1, an^1, n\#^1\}$
2	$\{s_1, s_3, s_4, s_5\}$	$\{\#v^1, vi^1, iv^1, vi^2, \#m^1, an^1, n\#^1\}$
1	$\{s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8\}$	$\{\#v^1, vi^1, iv^1, vi^2, ma^1, an^1, n\#^1\}$

Algorithm 1: GPC(D, q)

```
Data: $D$: dataset of strings; $q$: size of $q$-grams
Result: set of mutually disjoint clusters covering $D$
1  begin
2      $Clusters \leftarrow \emptyset$; // initialize the set of string clusters
3      $E \leftarrow D$; // initialize the set of eligible centers
4      while $E \neq \emptyset$ do // while non-clustered strings are left
5          $s \leftarrow$ random string from the set $E$ of eligible centers;
6          $P \leftarrow P(s, q)$;
           // compute the proximity graph
7          $PG[1..|P|]$ : empty array of neighborhoods; // initialize the proximity graph
8          $PG[|P|] = \{s\}$; // neighborhood of $P$ for $\tau = |P|$ is $\{s\}$
9          for $\tau = |P| - 1$ to 1 do // for each overlap threshold
10             $P \leftarrow P_c(P[\tau + 1], q)$; // $P$ is center of previously computed neighborhood
11             $PG[\tau] \leftarrow N(D, P, \tau) \cup PG[\tau + 1]$; // $\tau$-neighborhood of $P$ in $D$
12         $b \leftarrow$ cluster border (rightmost index of the longest horizontal line in $PG$);
           // update clusters and eligible centers
13         $Clusters \leftarrow Clusters \cup \{PG[b]\}$; // add new cluster
14         $E \leftarrow E \setminus PG[b]$; // clustered strings not eligible as centers
       // merge overlapping clusters
15     while $\exists C_i, C_j \in Clusters : C_i \neq C_j, C_i \cap C_j \neq \emptyset$ do
16         $Clusters \leftarrow Clusters \setminus \{C_i, C_j\} \cup \{C_i \cup C_j\}$;
17     return $Clusters$;
18 end
```

Fig. 2. GPC Clustering Algorithm

centers. When the set of eligible centers is empty, all strings are clustered. The resulting clusters may overlap. In order to get a hard clustering, overlapping clusters are merged.

GPC uses the proximity graph to detect cluster borders. The cluster border is a specific overlap threshold τ. Intuitively, the overlap is decreased until decreasing it further does not increase the neighborhood size. This corresponds to a *horizontal line* in the proximity graph. If there are multiple horizontal lines, the cluster border is defined by the longest one; between multiple horizontal lines of the same length the rightmost horizontal line is chosen. The cluster border is the overlap threshold at the right endpoint of the longest horizontal line.

Example 4. Consider the proximity graph $PG(s, D, q)$ illustrated in Figure 1(b). The longest horizontal line is given by $|N_3| = |N_4| = |N_5|$. Therefore, the cluster border is $b = 5$ and the cluster is $\{vivian, vivien, vivyan\}$.

2.3 State-of-the-Art Proximity Graph Computation

In Algorithm 1, the proximity graph is computed in the innermost loop in Lines 1–1. The critical operation in this loop is the computation of the neighborhood $N(D, P, \tau)$. The straightforward algorithm computes the overlap between P and each string in the dataset D and selects the strings with overlap at least τ. This approach is too expensive, since the neighborhood must be computed $|P|$ times for each proximity graph, and the number of proximity graphs is proportional to the number of strings in the dataset.

The state-of-the-art algorithm by Mazeika and Böhlen [1] uses an inverted list index and sampling. An inverted list $L(\kappa, D) = (s \mid s \in D, \kappa \in P(s,q))$ of a q-gram κ is the list of all strings $s \in D$ that contain κ. The inverted list index $LIndex(D, q)$ is the array of all inverted lists of the q-grams that exist for D.

Example 5. The inverted list index for $D = \{s_1, s_2, \ldots, s_8\} = \{vivian, adriana, vivien, marvin, vivyan, manuel, jeanne, clive\}$ consists of 36 inverted lists. The inverted lists of the q-grams in $P = \{\#v^1, vi^1, iv^1, vi^2, ya^1, an^1, n\#^1\}$ are

$$\begin{array}{ll} L(\#v^1, D) = (s_1, s_3, s_5) & L(ya^1, D) = (s_5) \\ L(vi^1, D) = (s_1, s_3, s_4, s_5) & L(an^1, D) = (s_1, s_2, s_5, s_6, s_7) \\ L(iv^1, D) = (s_1, s_3, s_5, s_8) & L(n\#^1, D) = (s_1, s_3, s_4, s_5) \\ L(vi^2, D) = (s_1, s_3) & \end{array}$$

A string is in the neighborhood $N_\tau = N(D, P, \tau)$ of profile P if it has τ q-grams in common with P. Thus all strings that are in the intersection of the inverted lists of τ q-grams of P are in the τ-neighborhood N_τ. In order to get *all* strings in N_τ, the inverted lists of all subsets of size τ of P must be intersected.

With $\{\kappa_1, \kappa_2, \ldots, \kappa_\tau\} \subseteq P$ we denote a subset of τ q-grams of P. The number of subsets of size τ of P is $\binom{|P|}{\tau}$, and we enumerate them as $\{\kappa_1^i, \kappa_2^i, \ldots, \kappa_\tau^i\}$, $1 \le i \le \binom{|P|}{\tau}$. The τ-neighborhood $N(D, P, \tau)$ is computed as the union of the intersected inverted lists for all subsets of size τ of P:

$$N(D, P, \tau) = \bigcup_{i=1}^{\binom{|P|}{\tau}} (L(\kappa_1^i, D) \cap L(\kappa_2^i, D) \cap \ldots \cap L(\kappa_\tau^i, D))$$

Obviously, it is not feasible to compute a combinatorial number of intersections. The state-of-the-art algorithm by Mazeika and Böhlen [1] uses sampling to deal with the high runtime complexity. Instead of computing the intersection for all subsets, the intersection is computed only for a sample of S subsets. The pseudo-code is shown in Algorithm 2.

Algorithm 2: PG-S(LIndex, S, s, q)

```
Data: $LIndex$: array of inverted lists, $S$: sample size, $s$: string; $q$: size of $q$-grams
Result: $PG$: array of neighborhoods
1  begin
2      $P \leftarrow P(s, q)$;
3      $PG[1..|P|]$ : empty array of neighborhoods; // initialize the proximity graph
4      $PG[|P|] = \{s\}$; // neighborhood of $P$ for $\tau = |P|$ is $\{s\}$
5      for $\tau = |P| - 1$ to 1 do
6          $P \leftarrow P_c(P[\tau + 1], q)$; // $P$ is center of previously computed neighborhood
7          for $i = 1$ to $S$ do
8              Generate a random $\tau$-subset $\{\kappa_1, \ldots, \kappa_\tau\} \subset P$;
9              $PG[\tau] \leftarrow PG[\tau] \cup (LIndex[\kappa_1] \cap \cdots \cap LIndex[\kappa_\tau])$;
10         $PG[\tau] \leftarrow PG[\tau] \cup PG[\tau + 1]$; // $\tau$-neighborhood for $P$ in $D$
11     return $PG$;
12 end
```

Fig. 3. PG-S: Proximity Graph Computation with Inverted Lists and Sampling

Sampling introduces an error, and the neighborhood is only approximated. Figure 1(c) shows an approximate proximity graph that results from the sampling approach; it is different from the exact graph in Figure 1(b).

Example 6. We continue Example 5 and compute the neighborhood $N(D, P, 3)$ for $P = \{\#v^1, vi^1, iv^1, vi^2, ya^1, an^1, n\#^1\}$ and sample size $S = 2$. We need to find the strings that contain at least 3 q-grams of P. First, we pick two random subsets of size τ of P and get $\{\#v^1, iv^1, ya^1\}$ and $\{vi^1, iv^1, an^1\}$. Then we compute the intersections of the inverted lists, $L(\#v^1, D) \cap L(iv^1, D) \cap L(ya^1, D) = \{s_1, s_3, s_5\} \cap \{s_1, s_3, s_5, s_8\} \cap \{s_5\} = \{s_5\}$ and $L(vi^1, D) \cap L(iv^1, D) \cap L(an^1, D) = \{s_1, s_3, s_5\} \cap \{s_1, s_3, s_5, s_8\} \cap \{s_1, s_2, s_5, s_6, s_7\} = \{s_1, s_5\}$. Finally, we compute $N_3 = \{s_5\} \cup \{s_1, s_5\} = \{s_1, s_5\}$.

3 Problem Definition

GPC is a clustering approach for strings that automatically detects the cluster borders. The brute force GPC algorithm intersects inverted q-gram lists to compute the cluster borders and requires a combinatorial number of intersections, making the exact computation of GPC infeasible. The state-of-the-art algorithm uses sampling to reduce the number of intersections and only approximates the GPC clusters, thus trading in quality for speed.

Our goal is to develop an *exact* algorithm for computing GPC clusters that is *efficient* and scales to large datasets with thousands of strings.

4 Efficient Proximity Graph Computation

In this section we present our solution for the exact proximity graph computation. We give a new data structure to manage inverted q-gram lists efficiently, we discuss an efficient algorithm to compute the center of a neighborhood, and finally introduce the PG-DS and PG-SM algorithms.

4.1 Initialization of String Profiles

In our algorithms we frequently need to access data for all q-grams of a profile, for example, their inverted lists. We assign consecutive IDs to strings and q-grams that allow us to perform this operation in constant time. For dataset D we initialize an array of size $|D|$ in which element i stands for string s_i and points to its profile. The profile is an array of q-gram IDs.

To produce this data structure, we scan the dataset and assign consecutive IDs to the strings. For each string s the profile $P(s, q)$ is computed. We maintain a dictionary of (κ^i, ID)-pairs that assigns consecutive IDs to q-grams κ^i and is implemented as a binary tree. We look up each q-gram of the newly computed profile in the dictionary. If the q-gram is there, we assign the respective ID, otherwise the q-gram is new, gets the next free ID, and is stored in the dictionary. The profile is stored as the array of its q-gram IDs. After the initialization the q-gram dictionary is no longer needed and can be dropped.

Example 7. We show the initialization step for dataset $D = \{s_1, s_2, \ldots, s_8\} = \{\textit{vivian}, \textit{adriana}, \textit{vivien}, \textit{marvin}, \textit{vivyan}, \textit{manuel}, \textit{jeanne}, \textit{clive}\}$. The dictionary maps the 36 distinct q-grams of D to IDs:

$$\begin{array}{cccccccccccccccc} \#v^1 & vi^1 & iv^1 & vi^2 & ia^1 & an^1 & n\#^1 & \cdots & nn^1 & ne^1 & e\#^1 & \#c^1 & cl^1 & li^1 & ve^1 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \cdots & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & \cdots & 30 & 31 & 32 & 33 & 34 & 35 & 36 \end{array}$$

We get the following data structure that associates the string IDs with the corresponding profiles of q-gram IDs (the string names shown below are not part of the data structure):

	String ID → *String Profile*		*String ID* → *String Profile*
vivian:	$1 \rightarrow (1, 2, 3, 4, 5, 6, 7)$	*vivyan*:	$5 \rightarrow (1, 2, 3, 21, 22, 6, 7)$
adriana:	$2 \rightarrow (8, 9, 10, 11, 5, 6, 12, 13)$	*manuel*:	$6 \rightarrow (16, 17, 6, 23, 24, 25, 26)$
vivien:	$3 \rightarrow (1, 2, 3, 4, 14, 15, 7)$	*jeanne*:	$7 \rightarrow (27, 28, 29, 6, 30, 31, 32)$
marvin:	$4 \rightarrow (16, 17, 18, 19, 2, 20, 7)$	*clive*:	$8 \rightarrow (33, 34, 35, 3, 36, 32)$

4.2 Computation of the Center Profile

Recall that the center of neighborhood N_τ is a profile with the K most frequent q-grams in N_τ, where K is the average length of the profiles in the neighborhood. In order to compute the center (see Algorithm 3), we maintain a min-heap of size K that stores (frequency, q-gram)-pairs. The top element is the pair with the minimum frequency value. In addition, we maintain an array of q-gram frequencies (histogram) and an array with the address of each q-gram in the min-heap (or -1 if the q-gram is not in the heap). The indices of both arrays are q-gram IDs, and we access the value for a specific q-gram in constant time.

First, we compute the size K of the center. Then we incrementally update the histogram with the frequencies of the q-grams that exist for the neighborhood N_τ. If the frequency of an updated q-gram is equal to or greater than the minimum frequency on the heap, the heap must be updated. If the q-gram is not yet in the heap, we add the q-gram and its frequency as a new element. Otherwise, we use the address array to find the q-gram on the heap in constant time; we update its frequency and heapify. We pop the top element if the heap grows larger than K. When all q-grams are processed, the heap stores the K most frequent q-grams of the neighborhood N_τ.

The heap operations *push*, *pop*, and *update* maintain the array of addresses whenever a heap element changes its position, is removed, or inserted. The *top* operation returns the minimum key, i.e., the minimum frequency of the heap.

Example 8. We continue Example 7 and compute the center of the neighborhood $N_3 = \{s_1, s_3, s_5\} = \{1, 3, 5\}$. The average profile size of the strings in the neighborhood is $K = 7$. The heap stores pairs $(\textit{hist}[\kappa], \kappa)$, where $\textit{hist}[\kappa]$ is the current frequency of κ in the histogram. After processing *vivian*, the top element of the heap is $(1, 1)$, the other elements are $(1, 2)$, $(1, 3)$, $(1, 4)$, $(1, 5)$, $(1, 6)$, $(1, 7)$. After *vivien*, the heap stores $(1, 15)$ (top), $(2, 4)$, $(1, 6)$, $(2, 1)$, $(2, 2)$, $(2, 3)$, $(2, 7)$. The final heap stores the K most frequent q-grams in the neighborhood N_3: $(1, 22)$ (top), $(2, 4)$, $(2, 6)$, $(3, 1)$, $(3, 2)$, $(3, 3)$, $(3, 7)$.

Algorithm 3: AdjustCenter(N, q)

```
Data: $N$: string neighborhood; $q$: size of $q$-grams
Result: center of neighborhood $N$
begin
    $minHeap$ : empty min-heap of (frequency, $q$-gram)-pairs;
    $\kappa_{max} \leftarrow \max(\bigcup_{s \in N} P(s,q))$; // find the $q$-gram with the maximum ID
    $addr[1..\kappa_{max}]$: $q$-gram addresses in $minHeap$; // initialize with $-1$'s
    $hist[1..\kappa_{max}]$: $q$-gram frequencies; // initialize with 0's
    $K \leftarrow \lfloor (0.5 + \sum_{s \in N} |P(s,q)|)/|N| \rfloor$; // compute the average size of the center
    foreach $s \in N$ do
        foreach $\kappa \in P(s,q)$ do // for each $q$-gram of $s$
            $hist[\kappa] \leftarrow hist[\kappa] + 1$; // increment the frequency of $\kappa$ in the histogram
            if $|minHeap| < K$ or $hist[\kappa] \geq top(minHeap)$ then
                if $addr[\kappa] = -1$ then // $\kappa$ not in the heap
                    $push(minHeap, (hist[\kappa], \kappa), addr)$;
                else // update frequency of $\kappa$ in heap
                    $update(minHeap, addr[\kappa], (hist[\kappa], \kappa), addr)$;
            if $|minHeap| > 0$ then $pop(minHeap, addr)$;
    return all $q$-grams stored in $minHeap$;
end
```

Fig. 4. AdjustCenter: Compute the Center of a Neighborhood

4.3 PG-DS Algorithm

In this section we present the PG-DS algorithm that computes the exact proximity graph and is based on the DivideSkip technique [2].

The input of PG-DS are the inverted list index $LIndex(D, q)$, the center string $s \in D$ for which the proximity graph should be computed, and the q-gram size (Algorithm 4). PD-DS computes the τ-neighborhoods by stepwise decreasing τ and calling DivideSkip at each step. The DivideSkip algorithm takes a set of inverted lists and a threshold τ and returns all string IDs that appear at least τ times in the inverted lists. In order to compute the τ-neighborhood of s, DivideSkip receives only the inverted lists of the q-grams of s, i.e., a subset $PIndex(D, s, q) \subseteq LIndex(D, q)$ such that $\forall \kappa \in P(s, q) : L(\kappa, D) \in PIndex$.

In the following paragraphs we give a short introduction to DivideSkip. DivideSkip is among the fastest algorithms for intersecting inverted lists and it combines MergeOpt and MergeSkip [2], both based on the Heap algorithm.

The Heap algorithm assumes all inverted lists to be ordered by string IDs. The heading (smallest) string ID of each inverted list is pushed to a min-heap. Further an array of counts is maintained, which stores the number of occurrences of each string ID. At each step, the algorithm pops the string with the smallest ID from the min-heap, inserts the next string on the corresponding inverted list to the heap, and increments the count of the popped string in the array of counts. A string is added to the result if its count meets the threshold τ. The runtime complexity of the algorithm is $O(m \log p)$, where p is the number of inverted lists and m is the total number of string IDs on these lists. For PG-DS, $p = |P(s, q)|$ and m is the number of strings in which the q-grams in $P(s, q)$ appear.

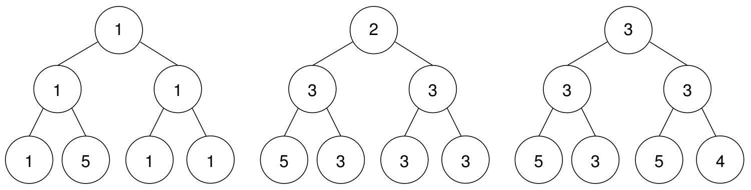


Fig. 5. Heap Structure for Computing N_3 Using MergeSkip in Example 9

MergeOpt improves over the Heap algorithm by processing the $\tau - 1$ longest inverted lists differently from the remaining shorter lists. On the shorter lists the Heap algorithm with threshold 1 is computed, resulting in a list of all string IDs that appear in these lists with their counts. For each candidate string on that list a binary search over the $\tau - 1$ long lists is executed to verify if the string appears at least τ times among all the lists.

MergeSkip extends the Heap algorithm in that it ignores irrelevant string IDs on the inverted lists. Instead of popping one element at a time, MergeSkip pops all elements from the heap that have the same string ID as the top element. If the number k of popped elements reaches τ, the string ID of this element is added to the result and the next k string IDs on the corresponding inverted lists are pushed to the heap. If $k < \tau$, the method additionally pops the $\tau - 1 - k$ smallest string IDs from the heap. For each inverted list of the $\tau - 1$ popped string IDs, the algorithm moves on to the smallest string ID on the list that is equal or greater than the current ID at the heap top and pushes it to the heap.

Example 9. We compute $N(D, P, 3)$ for $P = \{\#v^1, vi^1, iv^1, vi^2, ya^1, an^1, n\#^1\}$ using MergeSkip (see Example 5 for the inverted lists of P in D). Figure 5(a) shows the initial heap structure. First, we pop all the elements with value 1. Since 1 is popped 6 times, we add it to N_τ. The head string IDs of the corresponding

Algorithm 4: PG-DS(LIndex, s, q)

```
Data: $LIndex$: inverted list index; $s$: string ID; $q$: size of $q$-grams
Result: $PG$: array of neighborhoods
1  begin
2      $P \leftarrow P(s, q)$;
3      $PG[1..|P|]$ : empty array of neighborhoods; // initialize the proximity graph
4      $PG[|P|] = \{s\}$; // neighborhood of $P$ for $\tau = |P|$ is $\{s\}$
5      for $\tau = |P| - 1$ to 1 do
6          $P \leftarrow AdjustCenter(P[\tau + 1], q)$; // compute the center $P$ (see 4.2)
7          $PIndex[1..|P|]$ : empty inverted list index of $q$-grams in $P$
8          $i \leftarrow 0$; foreach $\kappa$ in $P$ do $PIndex[i{+}{+}] \leftarrow LIindex[\kappa]$;
9          $PG[\tau] \leftarrow DivideSkip(PIndex, \tau) \cup PG[\tau + 1]$; // $\tau$-neighborhood for $P$ in $D$
10     return $PG$;
11 end
```

Fig. 6. PG-DS: Proximity Graph Computation with DivideSkip

inverted lists are pushed to the heap (Figure 5(b)). We next pop 2, which appears only $k = 1$ times on the heap; thus we also pop the next $\tau - 1 - k = 1$ elements, i.e., one of the elements with string ID 3 (let us assume s_3 of the list $L(vi^1, D)$). We pop the $\tau - 1 = 2$ head elements of the corresponding lists, i.e., of $L(an^1, D)$ and $L(vi^1, D)$ (Figure 5(c)). Repeating these steps we finally get $N_\tau = \{s_1, s_3, s_5\}$.

DivideSkip partitions the inverted lists of $PIndex(D, s, q)$ into short and long lists. The number of long lists is l. MergeSkip is computed on the short lists to find the set of string IDs that appear at least $\tau - l$ times. Similar to MergeOpt, for each string in the result set of the short lists the algorithm does a binary search in the long lists. If the number of occurrences of the string among all the lists is at least τ, the string is added to the result. In our implementation of PG-DS, DivideSkip uses the empirical value for the number l of long lists that was proposed by Li et al. [2].

4.4 PG-SM Algorithm

In this section we present the PG-SM method, which uses a sort-merge join for computing the proximity graph.

Instead of the inverted list index, the PG-SM algorithm uses the sort-merge index $GIndex(D, q)$, an array of pairs $((\kappa, s)|\ \kappa \in P(s, q), s \in D)$ sorted by q-gram IDs. The size of $GIndex(D, q)$ is equal to $|\biguplus_{s \in D} P(s, q)|$.

Example 10. For the dataset $D = \{s_1, s_2, \ldots, s_8\} = \{$*vivian*, *adriana*, *vivien*, *marvin*, *vivyan*, *manuel*, *jeanne*, *clive*$\}$, index $GIndex(D, q)$ is shown below:

1	2	3	4	$\cdots$	53	54	55	56
$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\cdots$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$
$(\#v^1, s_1)$	$(\#v^1, s_3)$	$(\#v^1, s_5)$	(vi^1, s_1)	$\cdots$	$(\#c^1, s_8)$	(cl^1, s_8)	(li^1, s_8)	(ve^1, s_8)
$\Updownarrow$	$\Updownarrow$	$\Updownarrow$	$\Updownarrow$	$\cdots$	$\Updownarrow$	$\Updownarrow$	$\Updownarrow$	$\Updownarrow$
$(1, 1)$	$(1, 3)$	$(1, 5)$	$(2, 1)$	$\cdots$	$(33, 8)$	$(34, 8)$	$(35, 8)$	$(36, 8)$

In order to find strings that have τ q-grams in common with the center P, PG-SM (Algorithm 5) maintains a counter for each string in the array $ACounts$. For each q-gram κ of P, the algorithm does a binary search in $GIndex$. Once the position pos of κ is found, the algorithm looks for other copies of κ that immediately precede or succeed κ. For each copy of κ the count of the corresponding string in $ACounts$ is incremented by 1. Whenever the count of some string reaches τ, the string is added to the τ-neighborhood.

Example 11. We compute $N(D, P, 3)$ for $P = \{\#v^1, vi^1, iv^1, vi^2, ya^1, an^1, n\#^1\}$ using PG-SM. We search for the q-gram $\#v^1$ with ID 1 in $GIndex(D, q)$ and find it at position 3 in the pair $(1, 5)$ (see Example 10). We search for preceding/succeeding copies of $\#v^1$ and find the pairs $(1, 3)$, $(1, 1)$. We increment the counts of the strings $1, 3$, and 5. If the count of some string meets the threshold 3, we add it to the neighborhood. Repeating this procedure for all q-grams in P we get $N = \{s_1, s_3, s_5\}$.

Algorithm 5: PG-SM(GIndex, s, q)

```
Data: $GIndex$: sort-merge index of a dataset $D$; $s$: string ID; $q$: size of $q$-grams
Result: $PG$: array of neighborhoods
begin
    $ACounts[1..|D|]$: array of string frequences; // initialize string ID counts to 0's
    $P \leftarrow P(s, q)$;
    $PG[1..|P|]$ : empty array of neighborhoods; // initialize the proximity graph
    $PG[|P|] = \{s\}$; // neighborhood of $P$ for $\tau = |P|$ is $\{s\}$
    for $\tau = |P| - 1$ to 1 do
        $P \leftarrow AdjustCenter(P[\tau + 1], q)$; // compute the center $P$ (see 4.2)
        foreach $\kappa$ in $P$ do
            $pos \leftarrow BinarySearch(GIndex, \kappa)$; // get the position of $\kappa \in P$
            $TempSet \leftarrow \emptyset$; // initialize the temporary set of updated string IDs
            $i \leftarrow pos$;
            while $getGram(GIndex, i) = \kappa$ do
                $j \leftarrow getString(GIndex, i)$; $ACounts[j] \leftarrow AC[j] + 1$;
                if $ACounts[j] = \tau$ then $PG[\tau] \leftarrow PG[\tau] \cup \{s\}$;
                if $ACounts[j] = 1$ then $TempSet \leftarrow TempSet \cup \{j\}$;
                $i \leftarrow i - 1$;
            $i \leftarrow pos + 1$;
            while $getGram(GIndex, i) = \kappa$ do
                $j \leftarrow getString(GIndex, i)$; $ACounts[j] \leftarrow ACounts[j] + 1$;
                if $ACounts[j] = \tau$ then $PG[\tau] \leftarrow PG[\tau] \cup \{s\}$;
                if $ACounts[j] = 1$ then $TempSet \leftarrow TempSet \cup \{j\}$;
                $i \leftarrow i + 1$;
        foreach $j \in TempSet$ do $ACounts[j] \leftarrow 0$;
    return $PG$;
end
```

Fig. 7. PG-SM: Proximity Graph Computation with Sort-Merge Join

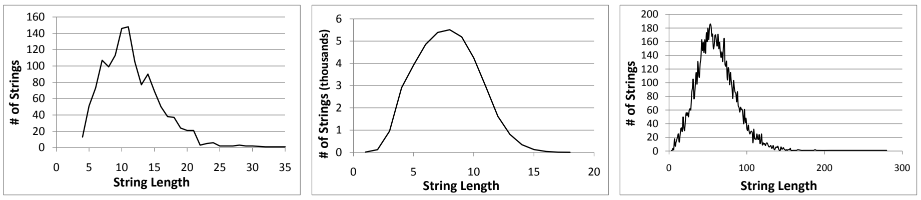
To avoid the initialization of all counts after each neighborhood computation, the algorithm maintains a temporary set of all strings, for which the counts were updated (*TempSet*). After the neighborhood computation, PG-SM only resets the counts of the strings in this set.

Let n be the size of the dataset D, $\bar{p} = |\bar{s}| + q - 1$ be the profile size of the average length string $|\bar{s}|$ in D. The time complexity of PG-SM for a string of length $\bar{s}$ is $O(\bar{p}^2 \log(n \cdot \bar{p}))$.

5 Experiments

Experimental Setup. We evaluate efficiency and effectiveness of our algorithms on three real-world datasets with different string length distributions (Figure 8). The Bozen dataset stores 1313 street names (4–35 characters); the Oxford dataset is a natural language collection of unique misspellings of the Oxford text archives with 39030 strings (1-18 characters); the DBLP dataset stores 10000 article titles with lengths up to 280 characters.

Clustering Efficiency. We compare the scalability of our algorithms (PG-DS and PG-SM) to the state-of-the-art algorithm PG-Sx, where x is the sample size. Figure 9 shows the runtime results for clustering each of the three datasets.



(a) Bozen Sreet Names (b) Oxford Misspellings (c) DBLP Paper Titles

Fig. 8. String Length Distributions

For the two larger datasets, Oxford and DBLP, we also show the computation times for subsets of different size. PG-DS is almost as fast as PG-S with the smallest sample size; only for the long strings of the DBLP dataset PG-S10 is significantly faster. PG-DS clearly outperforms PG-S for larger sample sizes in all settings. Note that PG-DS computes the exact GPC clusters, while PG-S10 only computes an approximation. Between our algorithms, PG-DS is consistently faster than PG-SM.

Proximity Graph Computation. The runtime for computing the proximity graph $PG(s, D, q)$ depends on the length of the center string s. We group the centers by their length and measure the average runtime of the proximity graph computation for each group. Figure 10 compares our exact algorithms with PG-S for different sample sizes. The runtime of PG-S quickly increases with the sample size, and only PG-S10 is comparable to our PG-DS and PG-SM algorithms.

Quality of the Proximity Graph. We measure the clustering quality using the established Normalized Mutual Information (NMI) [3], an information theoretic similarity measure between two clusterings. NMI is 1 for identical and zero for very different clusterings. The quality of a clustering is defined as its similarity to the ground truth, which is the exact GPC clustering in our experiments.

Figure 11 shows the tradeoff between the clustering quality and the runtime of GPC with sampling (PG-S) for different sample sizes. For the Bozen and Oxford datasets the smallest sample size gives only moderate results; good results can only be obtained with large samples of size 50 or 100 that drastically increase the runtime. PG-S with sample size 10 gives good results for the DBLP dataset; the strings in this dataset are long and strings from different clusters are very far from each other, making clustering easy. Note that our algorithms compute the exact GPC clusters, thus NMI = 1 for PG-DS and PG-SM.

Lesson Learned. The experiments show that our exact algorithms scale to large datasets. PG-DS outperforms PG-SM in all settings and is the algorithm of choice. The state-of-the-art algorithm PG-S uses sampling to approximate the GPC clusters. Our experiments show that PG-S is much slower than PG-DS for reasonable sample sizes in most settings. If the sample size of PG-S is reduced

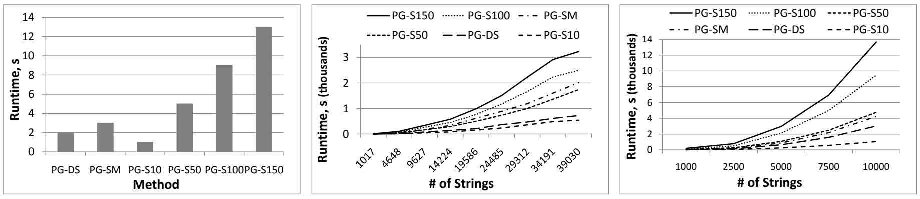

(a) Bozen Sreet Names (b) Oxford Misspellings (c) DBLP Paper Titles

Fig. 9. Clustering Runtime

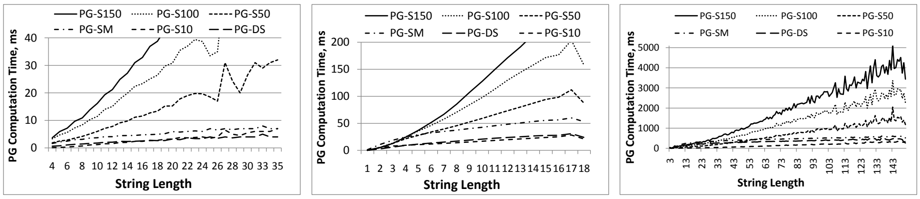
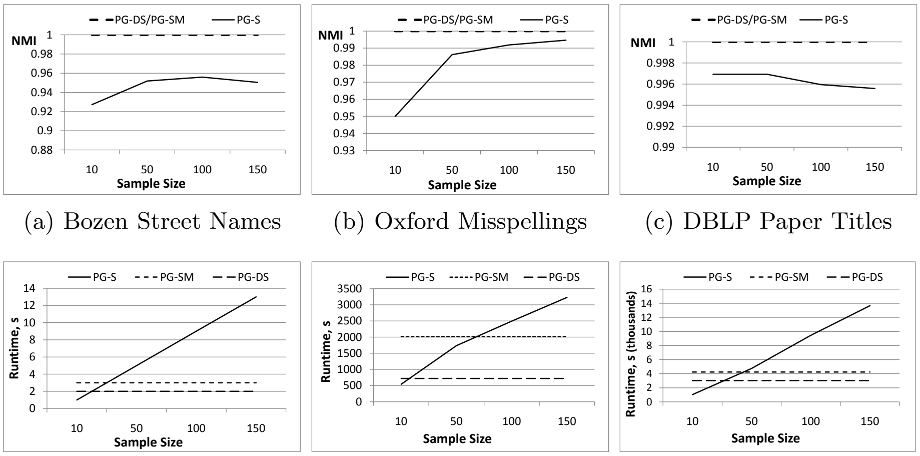

(a) Bozen Street Names (b) Oxford Misspellings (c) DBLP Paper Titles

Fig. 10. Runtime of Proximity Graph Computation


(a) Bozen Street Names (b) Oxford Misspellings (c) DBLP Paper Titles

(d) Bozen Street Names (e) Oxford Misspellings (f) DBLP Paper Titles

Fig. 11. (a)-(c) Quality vs. (d)-(f) Runtime of GPC with Sampling

to a small value such that PG-DS and PG-S have similar runtime, then the clustering quality of PG-S is typically low. PG-DS always computes the exact GPC clusters.

6 Related Work

Computing the similarity between strings is a problem that has received much attention from different research communities [4,5,6,7,8,9]. The q-gram distance [4,5], which is used for GPC [1], can be computed efficiently and it is an effective lower bound of the well-known string edit distance [10]. Fuzzy string matching techniques based on q-grams where used in data cleansing to find spelling mistakes and different syntactic representations [11,12].

Our PG-DS algorithm uses a divide-skip strategy to merge inverted q-gram lists efficiently. This strategy was introduced by Li et al. [5] and it solves the τ-occurrence problem; in our experiments we use the authors' implementation[2]. PG-SM uses a sort-merge join to solve the same problem. Sort-merge joins on q-grams have been used for approximate joins of strings [13] and XML trees [14].

7 Conclusion and Future Work

GPC is a clustering approach for strings that has successfully been used to cleanse non-dictionary strings [1]. Its distinguishing feature is the automatic cluster border detection, i.e., the number of clusters is not required as input. State-of-the-art GPC algorithms use sampling and only approximate clusters.

In this paper we addressed the problem of computing the *exact* GPC clusters efficiently and proposed the PG-DS and PG-SM algorithms. Both algorithms scale to datasets with thousands of strings. PG-DS, the faster of our algorithms, is faster than the state-of-the-art sampling algorithm, unless a very small sample size is used; small sample sizes give poor clustering results. To the best of our knowledge, we propose the first exact and scalable GPC algorithm.

We plan to further improve the scalability of GPC with pruning rules that allow to abort the cluster border detection early. Deciding the borders before the proximity graph is fully computed could substantially improve the runtime.

References

1. Mazeika, A., Böhlen, M.H.: Cleansing databases of misspelled proper nouns. In: CleanDB (2006)
2. Li, C., Lu, J., Lu, Y.: Efficient merging and filtering algorithms for approximate string searches. In: ICDE 2008: Proceedings of the 2008 IEEE 24th International Conference on Data Engineering, pp. 257–266. IEEE Computer Society, Los Alamitos (2008)
3. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge Univ. Press, Cambridge (2008)
4. Gravano, L., Ipeirotis, P.G., Jagadish, H.V., Koudas, N., Muthukrishnan, S., Pietarinen, L., Srivastava, D.: Using q-grams in a dbms for approximate string processing. IEEE Data Eng. Bull. 24(4), 28–34 (2001)
5. Li, C., Wang, B., Yang, X.: Vgram: Improving performance of approximate queries on string collections using variable-length grams. In: VLDB (2007)

[2] Flamingo package: `http://flamingo.ics.uci.edu`

6. Levenshtein, V.: Binary Codes Capable of Correcting Deletions, Insertions and Reversals. Soviet Physics Doklady 10, 707 (1966)
7. Landau, G.M., Vishkin, U.: Fast parallel and serial approximate string matching. J. Algorithms 10(2), 157–169 (1989)
8. Waterman, M.S., Smith, T.F., Beyer, W.A.: Some biological sequence metrics. Advances in Mathematics 20(3), 367–387 (1976)
9. Bilenko, M., Mooney, R., Cohen, W., Ravikumar, P., Fienberg, S.: Adaptive name matching in information integration. In: IEEE Intelligent Systems (2003)
10. Navarro, G.: A guided tour to approximate string matching. ACM Comput. Surv. 33(1), 31–88 (2001)
11. Chaudhuri, S., Ganti, V., Motwani, R.: Robust identification of fuzzy duplicates. In: International Conference on Data Engineering, pp. 865–876 (2005)
12. Xiao, C., Wang, W., Lin, X., Yu, J.X.: Efficient similarity joins for near duplicate detection. In: WWW 2008: Proceeding of the 17th International Conference on World Wide Web, pp. 131–140. ACM, New York (2008)
13. Gravano, L., Ipeirotis, P.G., Jagadish, H.V., Koudas, N., Muthukrishnan, S., Srivastava, D.: Approximate string joins in a database (almost) for free. In: VLDB 2001: Proceedings of the 27th International Conference on Very Large Data Bases, pp. 491–500. Morgan Kaufmann Publishers Inc., San Francisco (2001)
14. Augsten, N., Böhlen, M.H., Dyreson, C.E., Gamper, J.: Approximate joins for data-centric XML. In: ICDE, pp. 814–823. IEEE, Los Alamitos (2008)

Concurrency and Replica Control for Constraint-Based Database Caching

Joachim Klein

Databases and Information Systems, Department of Computer Science,
University of Kaiserslautern, Germany
jklein@cs.uni-kl.de

Abstract. Efficient and dynamic reallocation of data is a major challenge of distributed data management, because current solutions require constant monitoring and manual adjustment. In contrast, future solutions should provide autonomic mechanisms to achieve self-tuning and exhibit high degrees of flexibility and availability. Constraint-based database caching (CbDBC) is one of the most ambitious approaches to reach these goals, because it is able to dynamically respond to workload changes and keep subsets of data near by the application. In turn, caching of data always generates replicas whose consistency needs to be controlled—for reasons of data independence, transparent for both application and underlying DBMS. Hence, such a task can best be approached by a middleware-based solution.

This paper discusses challenges arising when distributed replicas are synchronized within CbDBC. We compare proposals using eager and lazy update propagation and review their feasibility within middleware-based realizations. Because constraints have to be maintained by the cache, they restrict the implementation of concurrency control mechanisms. Therefore, we explore, as a novel contribution, the far-reaching influence of these constraints.

1 Motivation

Similar to concepts used for *Web Caching*, database caching keeps subsets of records close to applications, which allows local and, hence, faster execution of declarative queries. In contrast to Web caching only supporting ID-based queries, database caching services set-oriented requests and must, therefore, verify that the predicates used by SQL queries can be evaluated, i.e., that their *predicate extensions* [12] are contained in the cache. To this end, constraint-based database caching (CbDBC) uses simple constraints (cp. Section 2), which need to be fulfilled at any time and allow to decide whether or not a predicate extension is completely kept. Many database vendors have extended their systems with similar but less flexible ideas [1,2,21]. The most important competitor approach uses materialized views to determine the predicate completeness of records cached [16]. A big challenge of these approaches is replica control, because caching of data always implies the existence of distributed replicas. In addition, database caching has to guarantee transaction properties, so that the employed concurrency control mechanism is of special importance. Because database caching has to solve exactly the problems occurring in (partially) replicated environments, the research results of this area are used as a starting point to choose an appropriate solution for CbDBC. Section 3 inspects the results and clarifies which methods can be used for replica control and concurrency control (CC) for database caching.

But, there is a major difference between caching and replication: the content of a cache is managed dynamically, which is a great advantage. The caching system can try to limit the number of replicas (regarding one data item) on its own, so that the number of caches that need to be updated remains small, even when many caches coexist. However, dynamic organization is the biggest problem. The constraints used to determine completeness become inconsistent, if updates are made. Therefore, applying an update requires additional refreshment or invalidation steps to guarantee consistency. Section 4 describes the problems arising for the constraint-based database caching approach.

Our CbDBC prototype ACCache [5] is realized as a middleware-based solution and, up to now, independent of a specific database system. We try to preserve this property and, hence, we explore the feasibility of various middleware-based approaches (cp. Section 5). But in doing so, we rely on the concurrency control of the underlying database system. First, we try to realize lazy update propagation being highly desired, before we explore eager approaches. Regarding schemes with lazy update propagation, we demonstrate that a middleware-based solution is only realizable providing limited functionality, so that just read-only transactions can be executed . Based on this observation, we explore eager approaches (cp. Section 5.2) which enable the cache to accelerate any read statement (also of writer transactions). Eager solutions, however, have to use RCC value locks (introduced in Section 4) to speed-up commit processing, which are not needed in lazy solutions.

The following section describes the constraint-based approach in more detail and repeats the most important concepts just for comprehension.

2 Constraint-Based Database Caching

A constraint-based database cache stores records of predicate extensions in so-called *cache tables*. The records are retrieved from a primary database system called *backend*. Each cache table T belongs to exactly one backend table T_B and, hence, their definitions are equivalent, except for foreign key constraints, which are not adopted. The tables and constraints maintained by a cache are represented as a so-called *cache group*.

A CbDBC system uses two different types of constraints to determine which predicate extensions are completely contained in the cache: Referential Cache Constraints (RCCs) and Filling Constraints (FCs). Both are defined using the fundamental concept of *value completeness*.

Definition 1 (Value completeness). *A value v is value-complete (or complete for short) in a column $T.a$ if and only if all records of $\sigma_{a=v}T_B$ are in T.*

An RCC $S.a \rightarrow T.b$ is defined between a source column $S.a$ and a target column $T.b$ (not necessarily different from $S.a$) to ensure value completeness in $T.b$ for all distinct values v in $S.a$. Please note, value completeness is just given for the target column. This allows, e. g., to execute an equi-join $S \bowtie_{a=b} T$ if value completeness is given for a value v in S (let us say for v in $S.c$), so that $\sigma_{S.c=v}(S \bowtie_{a=b} T)$ delivers the correct result. In the reverse case (value completeness is given for a value v in T), this join is could produce incomplete results[1].

Definition 2 (Referential cache constraint, RCC). *A referential cache constraint $S.a \rightarrow T.b$ from a source column $S.a$ to a target column $T.b$ is satisfied if and only if all values v in $S.a$ are value-complete in $T.b$.*

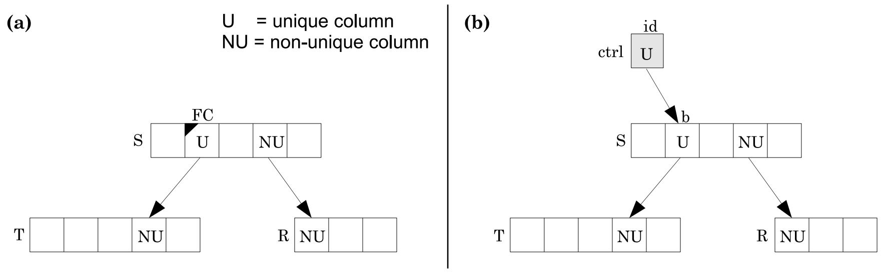


Fig. 1. The main components of a cache group and its internal representation

An FC is defined on a single column (e. g., $S.b$) and determines when a value v needs to be loaded completely. The loading is initialized as soon as a query refers to v explicitly (e. g., through $\sigma_{S.b=v}S$) and v is in a set of values to be loaded (called candidate values [12]). To implement the behavior of FC $S.b$, we internally use a so-called *control table* (*ctrl*) and an RCC *ctrl.id* $\rightarrow$ $S.b$ (cp. Figure 1). Conceptually, we put the value v in the id column of the control table. This violates RCC *ctrl.id* $\rightarrow$ $S.b$ and, hence, triggers maintenance, i.e., all records $\sigma_{b=v}S_B$ have to be made available. In doing so, new RCC source-column values in S arrive at the cache. This may violate outgoing RCCs which need to be satisfied again. In this way, depending on the value we put into the control table, the cache tables need to be filled-up in a consistent way.

Because of the special importance of values kept in an RCC source column, we denote such values as *control values* [15]. As illustrated, the presence of a control value (e. g., v) demands the availability of (recursively) dependent records. Hence, we denote this set of records as *closure* of the control value v.

Definition 3 (Closure of a control value). *Let v be a control value of RCC $S.a \rightarrow T.b$ and, thus, $I = \sigma_{a=v}T_B$ the set of records that have to be value-complete. The closure of v is the set of records $C(v) = I \cup C(v_i)$, $\forall v_i \in V(I)$, where $V(I) = (v_1, ..., v_n)$ denotes the control values included in I.*

[1] One reason that shows that an RCC is different from a foreign key.

Both constraint types (FCs and RCCs) are used to determine if a set of records currently stored in the cache is value-complete. Values of unique columns are implicitly, i.e., always complete. With help of this simple concept, it becomes possible to decide if predicates are completely covered by the cache and, hence, whether queries can be executed locally or not.

3 Preliminary Considerations

The main challenge regarding a replicated database environment is replica control. Typically used to control read-intensive workloads[2], our approach is based on a "read one replica write all (available) replicas" (ROWA(A)) schema. In the seminal paper of Gray et al. [8], replication techniques are classified by two parameters. The first parameter specifies where updates can be executed, at a primary copy or everywhere.

With database caching, caches temporarily hold data from a primary data source maintaining the consistency of all data items [12]. This so-called backend defines the schema that is visible to the user, whereas the caches as in-between components remain transparent. In contrast to replication, a primary copy approach fits into such a system architecture in a natural way, where the backend performs all updates and propagates them to the caches (if needed).

However, the given system architecture extremely complicates an *update-everywhere* solution. An important problem is that a cache cannot decide if an update violates constraints defined at the backend database, because it does not store any foreign-key constraints, check constraints, definitions for tables not present, triggers, or other information needed. All this meta-information had to be available to perform updates, so that cache maintenance could be done locally. In addition, *update everywhere* requires complex concurrency control or conflict resolution [8]; therefore, we strongly recommend the use of a primary copy approach where updates are always forwarded to the backend.

The second parameter introduced in Gray's paper [8] describes when replicas are refreshed, which can be done in *eager* or *lazy* fashion. In eager approaches, the changes of a transaction are propagated to all replicas before commit, whereas in lazy approaches the propagation may take place after commit. Because eager solutions delay transaction execution and lazy solutions have to deal with consistency problems, the most recent approaches (e. g., [3,11,14,17,22]) are designed as interim solutions, providing a well-defined isolation level and a so-called hybrid propagation, where, on the one hand, transactions accessing the same replicas do coordinate before commit (eager) and, on the other hand, successful commit of a replica is acknowledged to the client (lazy update propagation). In Section 5, we illustrate that, using database caching, it is possible to apply lazy update propagation without consistency problems. Hence, we can use an approach where commit is acknowledged to the client as soon as the transaction updates are committed at the backend database, while caches are updated lazily.

[2] This can be generally assumed in scenarios where database caching takes place.

However, update propagation must be combined with an adequate CC mechanism [17,22]. As the most important requirement when choosing or rather developing a tailor-made CC policy for CbDBC, the chosen mechanism should preserve a *caching benefit*, i. e., the performance gained from local query evaluation should not be outbalanced by cache maintenance. If the approach needs to access remotely maintained caches or the backend to perform read statements, the caching benefit will be compromised. For that reason, read accesses should never be blocked, e. g., to acquire distributed read locks as needed in a distributed two-phase locking (D2PL) approach. Another main problem for CC is that database caching is designed to cache data near to applications and, hence, caches are often allocated far away from the original data source, only reachable via wide-area networks. For that reason, a comparatively high network latency has to be anticipated to send CC messages (ca. 50–200 ms) and, thus, they must be avoided if possible.

If we scan recent research for CC approaches that fulfill these basic requirements, we find that only optimistic CC schemes and approaches using snapshot isolation (SI)[3] [4,7] as its isolation level are sufficient (cp. Section 5). Another possibility is to allow inconsistencies [9,10], but, first, the level of inconsistency needs to be defined by developers within SQL statements and, second, such cache systems do not scale if a high isolation level is required.

The preceding discussion clarifies that the basic assumptions and requirements of CbDBC dramatically decrease the number of viable approaches (for replica control and concurrency control). Moreover, it should be easy to combine the chosen approach for replica control with CC. Hence, our solutions provide SI which facilitate a simple integration with eager and lazy update propagation schemes.

In the following section, we examine the specific challenges that need to be solved if we implement update propagation, i. e., from the backend to the caches involved, within CbDBC.

4 Update Propagation

The process of propagating updates consists of three main tasks: gathering changed records (capture), identifying and informing the caches which are affected by changes (distribution), and accepting changes at the cache (acceptance).

Capture. ACCache can be used on top of any relational database system and relies just on the SQL interface to implement its functionality. Provided by most database systems [13,18,19,20], triggers or appropriate capture technology for changed data is thus necessary. Eager approaches (cp. Section 5.2), therefore, have to gather all changes of a transaction before commit. Lazy approaches, in turn, allow to collect changes after commit. Such an approach can be handled

[3] SI is a multi-version concurrency control mechanism presenting to a transaction T the DB state committed at $EOT(T)$. It does not guarantee serializable execution, but it is supplied by Oracle and PostgreSQL for "Isolation Level Serializable" or in Microsoft SQL Server as "Isolation Level Snapshot".

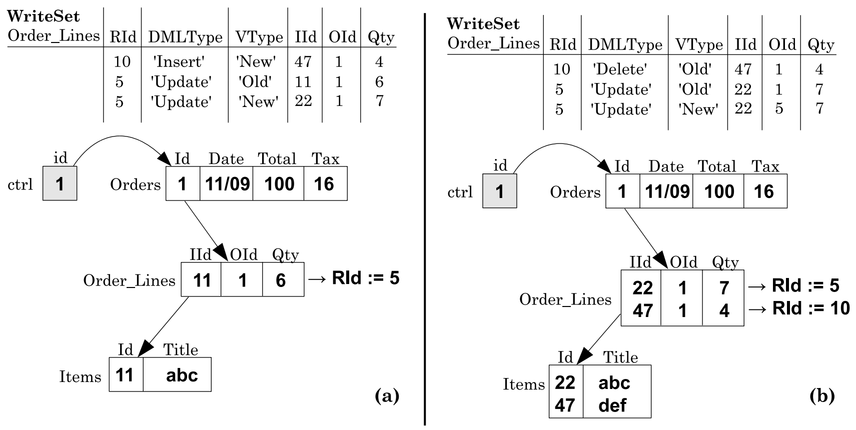
much more efficiently by log-sniffing techniques, which may be even processed concurrently. The collected data must be provided as a *write set (WS)* that includes at least the following information for each transaction: transaction id (*XId*), begin-of-transaction (*BOT*) timestamp, end-of-transaction (*EOT or commit*) timestamp, and all records changed. Furthermore, each record is described by an identifier (*RId*), the type of values included (*VType* := new values or old values), the type of DML operation (*DMLType* := insert, update, or delete), and the values themselves. For each update, two records (with old and new values) are included.

Distribution. For each record in a WS, we need to decide whether or not some of the connected caches have to be informed about the change. For this task, ACCache provides a dedicated service (Change Distributor, CD) running at the backend host. Depending on the meta-information maintained, a fine-grained or just a coarse-grained solution is possible. We distinguish the following levels of meta-information to be maintained:

– *Cache Information.* In this case, CD only knows that connected caches exist. Therefore, its only option is to ship the whole WS to all of them. In most cases, this level is not recommended, because, typically, just a few tables of a schema are of interest and, hence, appear in a cache.
– *Table Information.* This level additionally keeps for each cache the names of the cached tables. Therefore, selective shipment of the changed records is possible.
– *Cache Group Information.* It expands table information by storing all cache group definitions at the backend. Providing no additional help for the CD service, all changed records still have to be shipped to a cached table. However, we differentiate between this level and table information, because the backend can effectively assist loading policies using this additional information (cp. *prepared loading* in [15]).
– *Perfect Information.* In this case, CD maintains a special hash-based index to determine if a record is stored in a cache or not. With this support, it is possible to check each record of the WS and ship just the currently stored ones.

Obviously, perfect information enables a fine-grained selection of the records that need to be shipped to a cache. But the meta-information maintained must be continuously refreshed and needs to be consistent to ensure correctness. In summary, perfect information unnecessarily stresses the backend host and, therefore, we suggest using table information or cache group information.

Acceptance. The most critical part of CbDBC is the dynamic and concurrent acceptance of changes. Using "normal" replication, it is sufficient to reproduce the changes of the primary copy to refresh a replica. Within CbDBC, a change, for example, of value v to w may violate constraints and, therefore, changes in a cache must be hidden until all constraints are satisfied. Because we internally model FCs through RCCs (and control tables, cp. Section 2), we only need



WriteSet (a)

Order_Lines	RId	DMLType	VType	IId	OId	Qty
	10	'Insert'	'New'	47	1	4
	5	'Update'	'Old'	11	1	6
	5	'Update'	'New'	22	1	7

WriteSet (b)

Order_Lines	RId	DMLType	VType	IId	OId	Qty
	10	'Delete'	'Old'	47	1	4
	5	'Update'	'Old'	22	1	7
	5	'Update'	'New'	22	5	7

Fig. 2. Arrival of new control values (a), records losing their dependencies (b)

to observe RCC violations. The only situation, where RCCs can be violated, occurs when new control values reach the cache, i. e., during an update or insert. Figure 2a gives an example for this situation. The WS of table *Order_Lines* includes a new record with $RId = 10$ that inserts the control value $IId = 47$ and an update for the record with $RId = 5$ changing the control value 11 to 22. Hence, RCC $Order\_Lines.IId \rightarrow Item.Id$ is no longer satisfied and the closures of 22 and 47 need to be loaded first, before the WS can be accepted. In all other cases, the RCCs remain valid, but some records may be unloaded, if no incoming RCC implies their existence (cp. Figure 2b). Here, the delivered WS claims the deletion of record with $RId = 10$ and signals an update from $OId = 1$ to $OId = 5$ for the record with $RId = 5$. After acceptance of these changes, the closure of $Orders.Id = 1$ is empty and, thus, all records in the tables *Order_Lines* and *Items* should be unloaded.

Because the load of closures may be time consuming [15], commit processing may be considerably delayed. To overcome this problem, we use RCC value locks which indicate that, for a given RCC, some of the control values are not value-complete at the moment. This allows us to accept updates as soon as all locks are set (cp. Figure 3, where the same WS is applied as shown in Figure 2a).

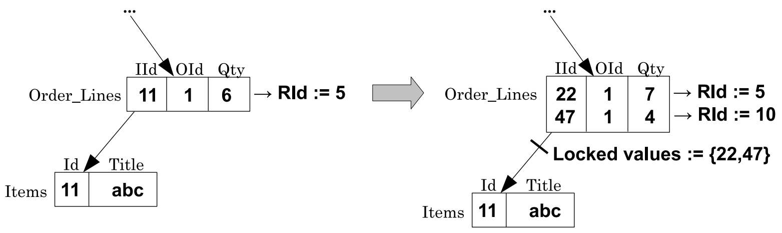


Fig. 3. Accepting of changes (WS from Figure 2a) using RCC value locks

These locks accelerate the processing of write sets, which is very important for eager concurrency control schemes. However, RCC locks constrain the execution of joins and, hence, using them with lazy approaches is not recommendable.

5 Concurrency Control

As described in Section 3, the most important requirement when choosing an appropriate CC mechanism is to preserve the caching benefit. Middleware-based solutions (like ACCache) are implemented on top of existing CC policies provided by the underlying database system and have to regard their special properties. Most restricting, a transaction T accessing these underlying systems has only access to the latest transaction-consistent state valid at $BOT(T)$. Hence, we denote this state as *latest snapshot*.

Observing current research activities, the simplest and, hence, most likely the best way to preserve the caching benefit is to allow read accesses without taking further actions, i. e., without retrieval of read locks, setting of timestamps, or collecting of read sets (e. g., for an optimistic CC policy). This has been possible since we know about the very powerful properties of SI, where reads are never blocked. It allows a cache to execute any read statement from any transaction without gathering information about elements read. However, to provide SI for database caching, the same snapshot has to be maintained for all statements of a transaction. Hence, this so-called *global snapshot* needs to be provided by a cache in combination with the backend.

This basic requirement can only be realized if either backend and caches provide always the latest snapshot (eager, cp. Figure 4a) or the caches have access to required snapshots at the backend (lazy, cp. Figure 4b).

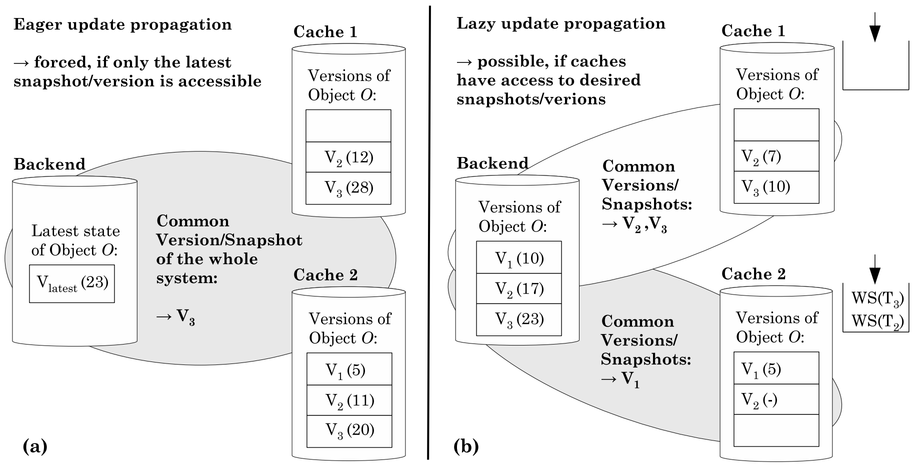


Fig. 4. Providing the same versions/snapshots either eager (a) or lazy (b)

In Figure 4, the point in time (represented through an integer value) when a version is locally committed is given in parentheses. Because cache and backend use their own *local* CC mechanisms, the timestamps assigned to the same version by both sides will differ. A cache is always supported by local CC mechanism providing SI and, hence, it maintains multiple versions. In the model for eager update propagation (given by Figure 4a), all caches have to accept a transaction's WS logically at the same time. During lazy update propagation (reconsider Figure 4b), caches maintain a queue of WSs that have to be applied in FIFO (first in first out) order. The queue of `Cache1` is currently empty and `Cache2` has to accept at first $WS(T_2)$ and after that $WS(T_3)$.

In all further explanations, we mark a read-only *user transaction* T_j with a subscript *sr* (e. g., T_{sr1}), if it executes just a single read, and with *mr*, if the transaction consists of multiple read statements. Write transactions are not differentiated any further.

Each *user transaction* is executed by a cache C_i and the backend where C_i is the cache that took control over the user transaction. Hence, for each user transaction T_j, the cache maintains a *cache transaction* T_j^{ca} to access its local data source (i. e., the cached data) and a *backend transaction* T_j^{be} to access the backend data. A cache transaction or a backend transaction not initiated by the user is simply marked with its purpose (e. g., T_j^{load} is used to load new cache contents).

Regarding correctness, it is sufficient to prove that each user transaction T_j realized with the aid of T_j^{ca} and T_j^{be} accesses the same snapshot S_i, because all changes are synchronized by the backend database.

5.1 Lazy Update Propagation

To allow lazy schemes, the only solution is to keep a backend transaction T^{load} open that allows reading the latest snapshot provided by the cache. Regardless of the problems of realizing a user transaction (i. e., commit cannot be processed without losing the link to the right snapshot), the *refresh* of such connections (i. e., switching to the next transaction representing a new snapshot) is critical. In addition, T^{load} must be used to execute read statements of user transactions that accesses the backend to reach the correct snapshot and because the commit of T^{load} is not permitted, the cache can just execute read-only user transactions.

Assume a cache retains a backend transaction T_1^{load} that reads a snapshot representing the state before transactions T_1 and T_2 are finished (cp. Figure 5a). To refresh T_1^{load} (e. g., after a short period of time), the cache creates a new transaction T_2^{load} representing the state after commit of T_1 and T_2. Given that the cache can arrange T_2^{load} (i. e., it can determine that BOT(T_2^{load}) $>$ EOT(T_1) and BOT(T_2^{load}) $>$ EOT(T_2)), the cache has to accept the changes of WS(T_1) and WS(T_2) (e. g., through a cache transaction T_1^{accept}), before switching to T_2^{load} is possible (cp. Figure 5b).

In addition, while changes are accepted, new records need to be loaded (cp. Section 4). The loading is still performed by T_1^{load} and, thus, newly loaded records can recursively be affected by changes within WS(T_1) and WS(T_2). Hence, each record loaded must be checked against WS(T_1) and WS(T_2) to ensure that all

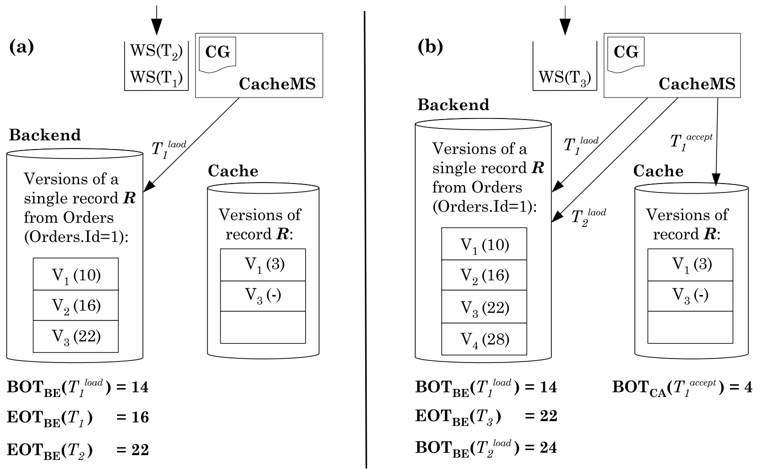


Fig. 5. Refreshing of T_1^{load} to T_2^{load}

changes get accepted correctly. Only if all changes within WS(T_1) and WS(T_2) have been processed completely or the loading over T_1^{load} gets shortly suspended, T_1^{accept} can be committed and T_2^{load} can be used for further processing. T_1^{load} is released as soon as no user transaction requires it anymore (i. e., if just user transactions T_j with BOT(T_j) > BOT(T_2^{load}) are executed by a cache).

Absolutely impossible is the usage of databases that apply pessimistic CC. The retained transaction will cause deadlocks and after an induced abort the state needed is no longer accessible.

Single-read transactions. To overcome the problem of accessing the same snapshot at cache and backend, we can try to allow only single-read transactions at the cache. We have to limit transactions to execute just one read statement, because further statements may need backend access. This approaches restrict the usage of the cache but, even in that case, lazy update propagation cannot be realized, as our following example shows.

Figure 6 illustrates the situation mentioned before. `Cache1` provides the transaction-consistent state (snapshot) before T_1 was committed, because the WS of T_1 is still available in the queue of WSs to be processed. For that reason, the transactions T_{sr6} and T_{sr7} are logically executed before T_1. `Cache2` has already applied WS(T_1) and, hence, T_{sr9} is after T_1 and before T_2. Assuming that backend and caches provide serializability for their local transactions, the transactions are also globally serializable, because writers are synchronized at the backend. This concept appears to be realizable in a simple way. Caches could be lazily refreshed by accepting the WSs delivered as explained in Section 4. However, global serializability is only guaranteed if caches always provide a transaction-consistent state for its locally executed transactions.

The following example clarifies that this is not possible if only the latest snapshot is accessible at the backend, because, then, necessary loading operations (e. g.,

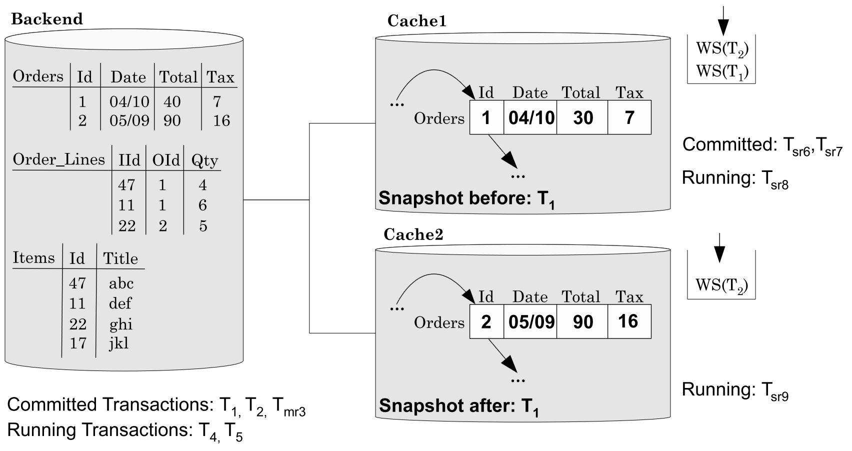


Fig. 6. Caches only executing single-read transactions

triggered through the standard filling behavior or during update acceptance) access different snapshots.

Considering Figure 7, we assume that the transaction T_1 increases the total amount of order 1 from 30 to 40 and of order 2 from 70 to 90. If we sum up the total amounts of order 1 and 2 before T_1 starts, we see a consistent state of 100; after commit of T_1 the sum of 130 is correct. The cache shown in Figure 7 has already loaded order 1 and starts accepting changes in the WS of T_1. Hence, at that moment, it provides the state before T_1. In this state, two single-read transactions run before the acceptance of WS(T_1) are finished. The first one, T_{sr1}, selects the order 2 that is not kept by the cache but triggers the loading of it. We assume that the loading occurs immediately and loads order 2, but now, the loading operation accesses the snapshot after T_1, because this is the latest snapshot and T_1 already committed at the backend. If the second transaction T_{sr2} now sums up the total amount of order 1 and 2, the answer given by the cache is 120, which was never a consistent state at the backend.

As a result, we conclude that a middleware-based solution with lazy update propagation can only be implemented on top of databases providing SI. Middleware-based approaches cause a lot of limitations: Changes of multiple write sets must be accepted together, only one transaction can be kept to access the correct snapshot, concurrently loaded records need to be checked against write sets, and the cache can just perform reads of read-only transactions.

Therefore, to support all read statements (also of writer transactions) at the cache using a middleware-based solution, building an eager solution is mandatory.

5.2 Eager Update Propagation

It is well known that eager solutions do not scale well, but within database caching they have important advantages: As backend, they allow any kind of

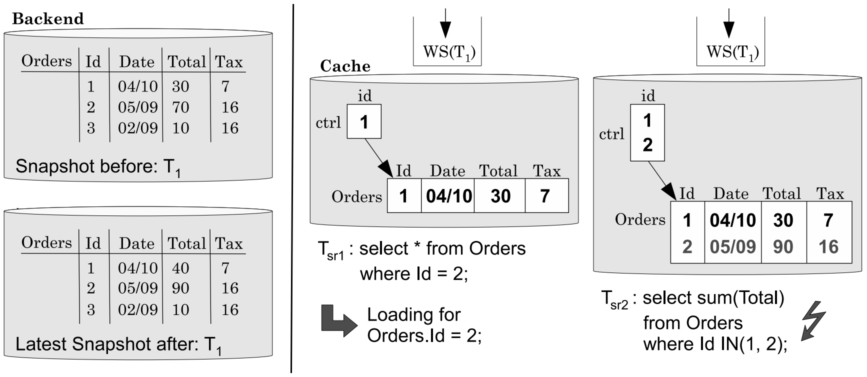

Backend
Orders | Id | Date | Total | Tax
1 | 04/10 | 30 | 7
2 | 05/09 | 70 | 16
3 | 02/09 | 10 | 16
Snapshot before: T_1
Orders | Id | Date | Total | Tax
1 | 04/10 | 40 | 7
2 | 05/09 | 90 | 16
3 | 02/09 | 10 | 16
Latest Snapshot after: T_1
Cache
WS(T_1)
id
ctrl 1
Orders | Id | Date | Total | Tax
1 | 04/10 | 30 | 7
T_{sr1} : select * from Orders where Id = 2;
Loading for Orders.Id = 2;
WS(T_1)
id
ctrl 1 2
Orders | Id | Date | Total | Tax
1 | 04/10 | 30 | 7
2 | 05/09 | 90 | 16
T_{sr2} : select sum(Total) from Orders where Id IN(1, 2);

Fig. 7. Loading operations construct an inconsistent state and cause chaos

database system providing SI[4] and their realization does not cause changes at the backend. In addition, if the locality at caches is very high[5], updates mostly affect only a very small number of caches (in the best case only one), so that commit processing performance is acceptable even when many caches co-exist. Subsequently, we explain the tailor-made commit procedure based on the well-known two-phase commit (2PC) protocol. It can be easily integrated with the update propagation described in Section 4 to reach a fully working concurrency control. However, beyond the limited scalability, the realization of eager approaches pose problems that impede the cache.

Commit Processing. A 2PC protocol synchronizes the replicas before commit and, thereby, guarantees that the backend and all caches provide only the latest snapshot to the user. The most ambitious challenge is to prepare the caches, so that the subsequent abort or commit message can be safely executed. But first of all, we respond to error processing which can be substantially simplified because cache databases do not need to be durable.

It allows us to commit transactions even if errors occur within the preparation phase at caches. After sending the WSs to the affected caches, the backend defines a period of time (timeout) in which the caches have to answer. If a cache signals a failed prepare or is not answering, it is invalidated. That means, all affected transactions are aborted. In the simplest case, a reinitialization is enforced (i. e., purging or restarting) to achieve a consistent state at failed caches. At the end, the transaction and all caches in the *prepared state* can commit.

[4] For eager methods, a database system using pessimistic CC could also serve as backend, in principle, but it may cause deadlocks that potentially affect performance. In addition, its use limits the level of isolation, because transactions running in the cache do not acquire read locks.

[5] Using database caching, the system can try to reach this state through an adaptive reorganization.

If the coordinator crashes, all running user transactions whose statements need to be redirected to the backend are automatically aborted. Transactions that were currently in the prepare phase are aborted after restarting the backend. If the backend is available again, normal processing can be continued without further adjustments. The last scenario nicely shows the improved fault tolerance of the global system, because the data kept by the caches remains accessible in case of a coordinator failure.

Preparation of Changes. When a cache receives a `Prepare` request of a user transaction (e. g., T_1), it has to process the corresponding $\mathrm{WS}(T_1)$ as fast as possible to shorten commit processing. Hence, loading of all records, needed to remedy violated RCCs (cp. Section 4), would imply excessive costs. Moreover, if an `Abort` message is received later, records were unnecessarily loaded. For that reason, we start a cache transaction T_1^{accept} that accepts the changes and assigns RCC value locks (cp. Figure 3) to all invalidated control values. As a result, the control values locked are invisible for probing, i. e., they can not be used to determine the completeness of records in target columns of RCCs checked. Furthermore, an RCC holding value locks can not be used to perform an equi-join.

As soon as all RCC value locks are applied, the cache sends the `Ready` signal to the backend and waits for the `Commit/Abort` message. If an `Abort` is received, the cache simply aborts T_1^{accept} and removes all RCC value locks previously assigned. After a `Commit` instruction, T_1^{accept} gets committed, too, and a loading process is initiated for each RCC value lock, which reconstructs value completeness for the control value and removes the lock.

Since the cache management system provides SI for its locally executed transactions, the changes and locks assigned through T_1^{accept} are only visible for local transactions T_i^{ca}, whose $\mathrm{BOT}(T_i^{ca}) > \mathrm{BOT}(T_1^{accept})$, so that transactions with $\mathrm{BOT}(T_i^{ca}) < \mathrm{BOT}(T_1^{accept})$ are not hindered.

Problems. Our proposal indicates how eager solutions should be designed for CbDBC. To cover the entire spectrum of approaches, we explored other opportunities as well [6], but all of them need to invalidate records or control values, which straiten the usage of cache contents.

But, furthermore, all proposals suffer from another important disadvantage: All participating instances of the system have to logically commit at the same time. If the backend would first commit (e. g., after having received all `Prepare` messages), the updates of a transaction T at the backend are immediately visible. But at the cache side, if the `Commit` instruction has not been received yet, T's snapshot is still present. Hence, a user transaction (e. g., T_2) initiating such a situation may access different snapshots (cp. Figure 8a). Of course, this problem can be simply avoided by redirecting requests from T_2 to the backend using T_2^{be} after the cache sends `Ready` to the backend. And after receiving `Commit`, a cache transaction T_2^{ca} can be initialized and used (cp. Figure 8b). But cache usage is again prevented for transactions initiated in the meantime.

In summary, eager approaches do not scale and, thus, they should not be used within wide-area networks.

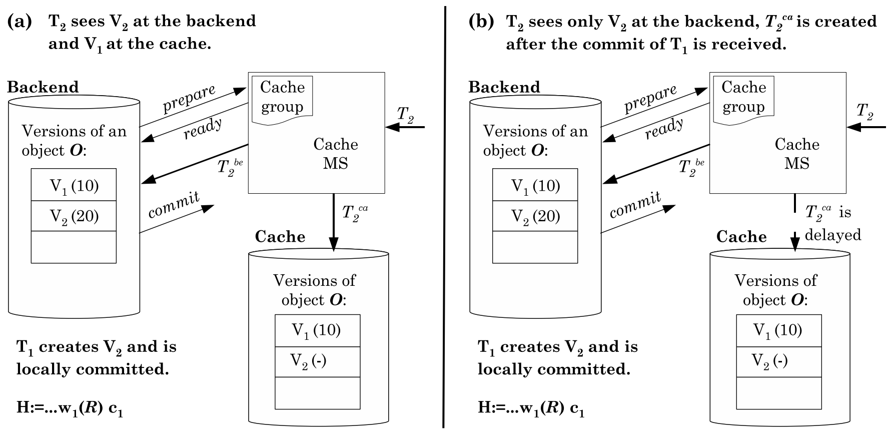


Fig. 8. T_2 can still access different snapshots (a) and is thus redirected (b)

6 Summary and Future Work

Using database caching, preserving the caching benefit is a superordinated requirement. In order to fulfill it, we can only use CC policies that allow to read the cached data without further coordination steps (e. g., accessing the backend database to acquire read locks). Multi-version concurrency control mechanisms providing SI (attached to the most recent database systems) offer almost perfect properties to realize this high concurrency level in database caching and especially in CbDBC. Middleware-based realizations are indeed implementable, but our observations clarify that they have some restrictions. The preferred lazy update propagation only supports read-only transactions and using eager update propagation, cache contents need to be temporarily locked. As a big drawback, backend systems using pessimistic CC policies may seriously affect transaction processing, because update acceptance in a cache triggers load operations that may cause deadlocks.

With help of our examination, we found out that an integration of lazy update propagation in a CC mechanism providing SI is realizable, but a middleware-based solution is challenging and shows restrictions, because backend transactions can only access the latest snapshot. Hence, we start implementing a closer integration of both CC mechanisms used at backend and cache, so that the cache has an influence of selecting the right snapshot for backend transactions.

References

1. Altinel, M., Bornhövd, C., Krishnamurthy, S., Mohan, C., Pirahesh, H., Reinwald, B.: Cache Tables: Paving the Way for an Adaptive Database Cache. In: VLDB Conf., pp. 718–729 (2003)
2. Amiri, K., Park, S., Tewari, R., Padmanabhan, S.: DBProxy: A Dynamic Data Cache for Web Applications. In: ICDE Conf., pp. 821–831 (2003)

3. Amza, C., Cox, A., Zwaenepoel, W.: Distributed Versioning: Consistent Replication for Scaling Back-end Databases of Dynamic Content Sites. In: Endler, M., Schmidt, D.C. (eds.) Middleware 2003. LNCS, vol. 2672, Springer, Heidelberg (2003)
4. Berenson, H., Bernstein, P.A., Gray, J., Melton, J., O'Neil, E.J., O'Neil, P.E.: A Critique of ANSI SQL Isolation Levels. In: SIGMOD Conference, pp. 1–10 (1995)
5. Bühmann, A., Härder, T., Merker, C.: A Middleware-Based Approach to Database Caching. In: Manolopoulos, Y., Pokorný, J., Sellis, T.K. (eds.) ADBIS 2006. LNCS, vol. 4152, pp. 184–199. Springer, Heidelberg (2006)
6. Braun, S.: Implementation and Analysis of Concurrency Control Policies for Constraint-Based Database Caching. Master's thesis, TU Kaiserslautern (2008) (in German), `http://wwwlgis.informatik.uni-kl.de/cms/fileadmin/users/jklein/documents/_2008_Braun_.DA.pdf`
7. Fekete, A.: Snapshot Isolation. In: Ency. of Database Systems, pp. 2659–2664 (2009)
8. Gray, J., Helland, P., O'Neil, P.E., Shasha, D.: The Dangers of Replication and a Solution. In: SIGMOD Conference, pp. 173–182 (1996)
9. Guo, H., Larson, P.A., Ramakrishnan, R.: Caching with "Good Enough" Currency, Consistency, and Completeness. In: VLDB, VLDB Endowment, pp. 457–468 (2005)
10. Guo, H., Larson, P.A., Ramakrishnan, R., Goldstein, J.: Relaxed Currency and Consistency: How to Say "Good Enough" in SQL. In: SIGMOD, pp. 815–826. ACM, New York (2004)
11. Holliday, J., Agrawal, D., Abbadi, A.E.: The Performance of Database Replication with Group Multicast. In: FTCS, pp. 158–165 (1999)
12. Härder, T., Bühmann, A.: Value Complete, Column Complete, Predicate Complete – Magic Words Driving the Design of Cache Groups. The VLDB Journal 17(4), 805–826 (2008)
13. IBM: InfoSphere Change Data Capture (2009), `http://www-01.ibm.com/software/data/infosphere/change-data-capture/`
14. Jiménez-Peris, R., Patiño-Martínez, M., Kemme, B., Alonso, G.: Improving the Scalability of Fault-Tolerant Database Clusters. In: International Conference on Distributed Computing Systems, p. 477 (2002)
15. Klein, J., Braun, S.: Optimizing Maintenance of Constraint-Based Database Caches. In: ADBIS, pp. 219–234 (2009)
16. Larson, P., Goldstein, J., Zhou, J.: MTCache: Transparent Mid-Tier Database Caching in SQL Server. In: ICDE Conf., pp. 177–189 (2004)
17. Lin, Y., Kemme, B., Patiño-Martínez, M., Jiménez-Peris, R.: Middleware-based Data Replication providing Snapshot Isolation. In: SIGMOD, pp. 419–430 (2005)
18. Microsoft Corporation: SQL Server 2008–Change Data Capture (2009), `http://msdn.microsoft.com/en-us/library/bb522489.aspx`
19. Oracle Corporation: Data Warehousing Guide–Change Data Capture (2009), `http://download.oracle.com/docs/cd/E11882_01/server.112/e10810.pdf`
20. The PostgreSQL Global Development Group: Postgresql 8.4.3 Documentation–Triggers (2009), `http://www.postgresql.org/files/documentation/pdf/8.4/postgresql-8.4.3-A4.pdf`
21. The TimesTen Team: Mid-tier Caching: The TimesTen Approach. In: SIGMOD Conf., pp. 588–593 (2002)
22. Wu, S., Kemme, B.: Postgres-R(SI): Combining Replica Control with Concurrency Control based on Snapshot Isolation. In: ICDE, pp. 422–433 (2005)

Exploiting Conflict Structures in Inconsistent Databases

Solmaz Kolahi and Laks V.S. Lakshmanan

University of British Columbia
{solmaz,laks}@cs.ubc.ca

Abstract. For an inconsistent database that violates a set of (conditional) functional dependencies, we define a basic conflict as a minimal set of attribute values, of which at least one needs to be changed in any attribute-based repair. Assuming that the collection of all basic conflicts is given, we show how we can exploit it in two important applications. The first application is cleaning the answer to a query by deciding whether a set of tuples is a possible answer, i.e., they are present in the result of the query applied to some minimal repair. We motivate an alternative notion of answer with a consistent derivation, which requires that the tuples are obtained through the same occurrences of attribute values in both the inconsistent database and the repair. The second application is cleaning data by generating repairs that are at a "reasonable" distance to the original database. Finally, we complement the above results and show that, if dependencies do not form a certain type of cycle, the cardinality of basic conflicts in any inconsistent database is bounded, and therefore it is possible to detect all basic conflicts in an inconsistent database in polynomial time in the size of input database.

1 Introduction

Dirt or inconsistency in data is a common phenomenon in many applications today. Reasons for inconsistency abound and include, among other things, error in data entry, inconsistency arising from collecting information from multiple sources with "conflicting" facts, or just the plain uncertain nature of the available data. Inconsistency in data is usually captured as violations of the constraints that the data is supposed to obey. Integrity constraints such as key and functional dependencies, or even statistical constraints about the data are prime examples of constraints that are useful in inconsistency management.

The two well-known approaches for exploiting constraints in managing inconsistency are *data cleaning* and *consistent query answering*. Data cleaning basically refers to minimally repairing or modifying the database in such a way that the (integrity) constrains are satisfied [11,9,25]. In consistent query answering [2,8,14,4,15], the goal is extracting the most reliable answer to a given query by considering every possible way of repairing the database. In both of these approaches, the notion of *minimal repair* is used: an updated version of the database that is minimally different and satisfies the constraints, which can be obtained by inserting/deleting tuples or by modifying attribute values.

name	*postalCode*	*city*	*areaCode*	*phone*
Smith	V6B	Vancouver	514	123 4567
Simpson	V6T	Vancouver	604	345 6789
Rice	H1B	Montreal	514	876 5432

(a)

name	*postalCode*	*city*	*areaCode*	*phone*
Smith	V6B	Vancouver	?	123 4567
Simpson	V6T	Vancouver	604	345 6789
Rice	H1B	Montreal	514	876 5432

name	*postalCode*	*city*	*areaCode*	*phone*
Smith	?	Montreal	514	123 4567
Simpson	V6T	Vancouver	604	345 6789
Rice	H1B	Montreal	514	876 5432

(b)

Fig. 1. (a) An inconsistent database and its basic conflicts (b) Two minimal repairs

In this paper, we consider inconsistent databases that violate a set of functional dependencies, or conditional functional dependencies, a recent useful extension to functional dependencies [19]. We focus on attribute-based repairs that are obtained by modifying attribute values in a minimal way. We are interested in two main problems. The first problem is *clean query answering* by identifying impossible portions of a query answer. More specifically, given a set of tuples in the answer, we would like to find out whether the set will appear in the result of the query on any repair of the inconsistent database. The second problem is data cleaning, specifically, finding repairs that have been obtained by making (close to) minimum changes to the input inconsistent instance.

We introduce the notion of *basic conflict* as a set of attribute value occurrences, not all of which can remain unchanged in *any* minimal repair. A basic conflict can be thought of as an independent source of inconsistency that needs to be resolved in *any* repair. We show that we can benefit from knowing basic conflicts both in data cleaning and in clean query answering.

Example 1. Figure 1(a) shows a database instance containing address information for a number of customers. We know, as a fact, that when the value of attribute *postalCode* is V6B, then *city* should be Vancouver. Also, if the value of attribute *areaCode* is 514, then *city* should be Montreal. These constraints can be expressed as conditional functional dependencies (see [19]). Notice that for customer Smith, there is a dependency violation with *areaCode* and *city*, and one of these values must be changed in any repair. Thus, the set of positions corresponding to the values of these attributes, shown with boxes, form a basic conflict. There is another source of inconsistency with the value of attributes *postalCode* and *areaCode* in the same tuple: they imply different cities for Smith, and therefore no repair can retain both of these values. Thus, the two shaded positions form a second basic conflict. Now consider the query: find postal codes of customers in the area code 514. Clearly, V6B is a wrong answer. It would be beneficial if we could detect this wrong tuple in the answer by propagating the basic conflicts. If we want to take the data cleaning approach instead and modify this database to make it consistent, we can pick any of the two repairs shown in Figure 1(b) that have changed at least one value in every basic conflict. □

Our solution to clean query answering is to make query answering "conflict-aware", by propagating the basic conflicts in an inconsistent database instance, to the answer of a query, and then check whether a set of tuples in the query answer is created by a conflicting set of values. If not, those tuples form a possible answer. For recording and propagating the information on the positions of basic conflicts we make use of annotations and use a mechanism similar to provenance management. More precisely, we annotate every data value in the inconsistent database with a unique annotation, say a natural number, and represent each basic conflict as a set of annotations. We define a simple annotated relational algebra that copies the annotations in the input instance to the tuples selected in the answer and records a light form of provenance for each tuple. We motivate the notion of *answers with a consistent derivation* as a restrictive alternative to possible answers that admits easier reasoning. Intuitively, a set of tuples in the answer of a query has a consistent derivation if they can appear in the answer of the query over some repair exactly through the same occurrences of attribute values of the inconsistent database. Clearly, any answer with a consistent derivation is a possible answer. We develop sufficient conditions that ensure a set of tuples has a consistent derivation (and therefore is a possible answer). For restricted classes of queries we show our condition is also necessary. We also show that for certain classes of queries, the notions of possible answer and answer with a consistent derivation coincide. It is worth pointing out two appealing features of our approach. First, it is compositional, in the sense that by having the annotated answer to a view, written as a query, we can reuse it for clean query answering over the view. Second, there is no extra overhead for processing queries using the annotated relational algebra if basic conflicts are identified ahead of time.

In addition to query answering, we take advantage of basic conflicts in data cleaning. To clean the data, there is usually a large number of minimal repairs to choose from. The quality of a repair is sometimes evaluated by a distance or cost measure that shows how far the repair is from the original inconsistent database. Finding minimum repairs w.r.t. such a cost measure is usually NP-hard, and we need techniques to generate repairs whose distance is reasonably close to minimum [9,11,20,29,16,25]. We show that if the collection of basic conflicts is given, then a minimum-cost or optimum repair would correspond to the minimum hitting set of the collection of basic conflicts. Using this, we can apply a greedy approximation algorithm for finding an approximate solution to the minimum hitting set, and produce a repair whose distance to the original database is within a constant factor of the minimum repair distance. This is under the condition that the cardinality of all basic conflicts is bounded.

For both of the above approaches to work, we need to have the complete collection of basic conflicts in the database. However, this may sound too ambitious as basic conflicts can come in unusual shapes, as shown in the next example.

Example 2. Consider the set of functional dependencies $\Sigma_1 = \{AB \rightarrow C, CD \rightarrow A\}$, and the database instance shown below.

A	B	C	D
a_1	b_1	c_1	
a_1	b_1		d_1
a_2		c_1	d_1

It is easy to see that no matter how the blank positions are filled, we cannot come up with a consistent instance. In other words, the values shown in the instance form a basic conflict. □

We next look at the complexity of detecting the collection of basic conflicts for an input inconsistent database. We present a sufficient condition on the cycles, that may exist in the set of functional dependencies, which guarantees an instance-independent bound on the cardinality of each basic conflict for any given input database. We therefore conclude that for this class of functional dependencies, it is possible to compute the collection of basic conflicts in polynomial time in the size of the input database.

Related Work. Attribute-based repairs have been studied by many both for query answering over inconsistent databases [9,20,31,24] and for generating repairs that are close to the inconsistent database with respect to a distance or cost measure [9,11,29,16,25]. Query answering over inconsistent databases has mostly focused on finding the certain answer: tuples that persist in the answer of the query over all repairs, which is usually done by query rewriting [2,21,32] or using logic programs [3]. In this paper, we consider detecting tuples that are in the answer to the query over some minimal attribute-based repair. In other words, we focus on possible answers. Possible query answering is a well-studied subject in incomplete databases (see, e.g., [1,23]).

The notion of conflicts has been used before in the context of inconsistent databases, but it mostly refers to a group of tuples that do not satisfy a key constraint or the general form of a denial constraint [15,9,4]. A natural and appealing alternative to tuple deletions for database repair is to change attribute values. Since we adopt this model in this paper, we use the notion of conflict as a set of attribute values. There is a similarity between a basic conflict and an incomplete database that does not *weakly satisfy* a set of dependencies, which has been studied in the context of incomplete databases (see [28,27]).

Annotations and provenance (or lineage) have previously been used for many problems, such as view maintenance [13], query answering over uncertain and probabilistic data [7,22], and consistent query answering over inconsistent databases with logic program [5,6]. Our work is different in that, through annotations and provenance, we would like to propagate conflicts among attribute values in a database that are not necessarily within a single tuple. The annotations that we use provide some sort of *where provenance* as introduced in [12].

The paper is organized as follows. In Section 2, we provide the necessary background, define the notion of basic conflict, and show how they relate to minimal repairs. We present the applications of basic conflicts in query answering and data cleaning in Sections 3 and 4. We discuss the complexity of detecting conflicts in Section 5. Finally, in Section 6, we present our concluding remarks. Some of the omitted proofs can be found in the full version of paper [26].

2 Inconsistent Databases

We denote a database schema by a set of relation names $S = \{R_1, \ldots, R_l\}$. A functional dependency (FD) over attributes of relation R_i ($sort(R_i)$) is an expression of the form $X \rightarrow Y$, where $X, Y \subseteq sort(R_i)$. An instance I_{R_i} of relation R_i satisfies $X \rightarrow Y$, denoted by $I_{R_i} \models X \rightarrow Y$, if for every two tuples t_1, t_2 in I_{R_i} with $t_1[X] = t_2[X]$, we have $t_1[Y] = t_2[Y]$. An instance satisfies a set of FDs Σ, if it satisfies all FDs in Σ. We say that an FD $X \rightarrow A$ is implied by Σ, written $\Sigma \models X \rightarrow A$, if for every instance I satisfying Σ, I satisfies $X \rightarrow A$. The set of all FDs implied by Σ is denoted by Σ^+. In this paper, we always assume that Σ is *minimal*, i.e., a set of FDs of the form $X \rightarrow A$ (with a single attribute on the right-hand side), such that $\Sigma \not\models X' \rightarrow A$ for every $X' \subsetneq X$, and $\Sigma - \{X \rightarrow A\} \not\models X \rightarrow A$. We usually denote sets of attributes by X, Y, Z, single attributes by A, B, C, and the union of two attribute sets X and Y by XY.

Conditional functional dependencies (CFDs) have recently been introduced as an extension to traditional functional dependencies [10,16,19,18]. CFDs are capable of representing accurate information and are useful in data cleaning. A CFD is an expression of the form $(X \rightarrow A, t_p)$, where $X \rightarrow A$ is a standard FD, and t_p is a *pattern tuple* on attributes XA. For every attribute $B \in XA$, $t_p[B]$ is either a constant in the domain of B or the symbol '_'. To define the semantics of CFDs, we need an operator $\asymp$. For two symbols u_1, u_2, we have $u_1 \asymp u_2$ if either $u_1 = u_2$ or one of u_1, u_2 is '_'. This operator naturally extends to tuples. Let I_R be an instance of relation schema R, and Σ be a set of CFDs over the attributes of R. Instance I_R satisfies a CFD $(X \rightarrow A, t_p)$ if for every two tuples t_1, t_2 in I_R, $t_1[X] = t_2[X] \asymp t_p[X]$ implies $t_1[A] = t_2[A] \asymp t_p[A]$. Like traditional FDs, we can assume that we deal with a minimal set of CFDs [10].

2.1 Repairs and Basic Conflicts

In this paper, we deal with inconsistent database instances that do not satisfy a set of dependencies. We consider repairs that are modifications of the database by changing a minimal set of attribute values. We therefore need a notion of *tuple identifier* with which we can refer to a tuple and its updated version. In a relational database, a tuple identifier can be implemented as a primary key whose values are clean. In general, we say that an attribute is *clean* if the values of this attribute never involve in any dependency violation. We also need a notion of *position* to refer to a specific attribute value for a given tuple identifier. For an instance I of schema $S = \{R_1, \ldots, R_l\}$, the set of positions of I is defined as $Pos(I) = \{(R_j, t, A) \mid R_j \in S, A \in sort(R_j)$, and t identifies a tuple in $I_{R_j}\}$. We denote the value contained in position $p = (R_j, t, A) \in Pos(I)$ by $I(p)$ (or $I(R_j, t, A)$). For two instances I and I' such that $Pos(I) = Pos(I')$, the *difference* between I and I' is defined as the set $Diff(I, I') = \{p \in Pos(I) \mid I(p) \neq I'(p)\}$.

```
**Algorithm** HittingSetRepair
**Input:** Instance $I$, FDs $\Sigma$, hitting set $H$ of $\Sigma$-$Confs(I)$.
**Output:** A minimal repair $I_H$ for $I$.
$I_H := I$;
$change := \emptyset$;
**while** there is an FD $X \rightarrow A \in \Sigma$ and tuples $t_1, t_2$ in $I_R$ such that $t_1[X] = t_2[X]$ and
$\{(R, t_i, B) \mid i \in [1,2], B \in XA\} \cap (H \setminus change) = \{(R, t_2, A)\}$ **do**
    $I_H(R, t_2, A) := I_H(R, t_1, A)$;
    $change := change \cup \{(R, t_2, A)\}$;
**while** $change \neq H$ **do**
    pick $p \in H \setminus change$;
    $I_H(p) := a$, where $a$ is a fresh value not in the active domain of $I_H$;
    $change := change \cup \{p\}$;
**return** $I_H$;
```

Fig. 2. Constructing a minimal repair for a minimal hitting set of Σ-*Confs*(I)

Let Σ be a set of (conditional) functional dependencies, and I be an instance of schema S that does not satisfy Σ, i.e., $I \not\models \Sigma$. Instance I' of S is an *attribute-based repair* for I if $Pos(I) = Pos(I')$ and $I' \models \Sigma$. We say that I' is a *minimal repair* if there is no repair I'' of I such that $Diff(I, I'') \subsetneq Diff(I, I')$.

Definition 1. *For a database instance I that does not satisfy a set of (C)FDs Σ, a set of positions $T \subseteq Pos(I)$ is a basic conflict if there is no repair I' of I such that $Diff(I, I') \cap T = \emptyset$, and no proper subset of T has this property. The collection of all basic conflicts for an instance I is denoted by Σ-$Confs(I)$.*

In this paper, we assume that there is no domain constraint for the attribute values in a database that could interact with (conditional) functional dependencies, e.g, constraints that put a restriction on the number of occurrences for each value. Moreover, the domain of each attribute is large enough that, there are always enough fresh values, not already in the active domain, to replace the existing values. With these assumptions, it is easy to see that once the collection of basic conflicts for an inconsistent database is given, then minimal repairs would exactly correspond to minimal hitting sets of this collection. Recall that for a collection of sets over the elements of a universe, a hitting set is a subset of the universe that intersects every set in the collection. A minimal hitting set is a hitting set such that none of its proper subsets is also a hitting set.

Theorem 1. *Let Σ be a set of (conditional) functional dependencies, and I be a database instance, then*

1. *for every minimal hitting set H of Σ-$Confs(I)$, there is a minimal repair I_H of I, such that $Diff(I, I_H) = H$.*
2. *for every minimal repair I' of I, $Diff(I, I')$ is a minimal hitting set for Σ-$Confs(I)$.*
3. *for a set of positions $P \subseteq Pos(I)$, there is a minimal repair I' for I with $Diff(I, I') \cap P = \emptyset$ if and only if for every conflict $T \in \Sigma$-$Confs(I)$, $T \not\subseteq P$.*

The chase procedure in Figure 2 shows how a repair I_H can be constructed for a given hitting set H of Σ-$Confs(I)$, when Σ is a set of FDs. After changing the value of positions that are forced by FDs, the other positions in the hitting set are updated with fresh values that are not in the active domain. The chase could be similarly done for a set of CFDs.

3 Conflict-Aware Query Answering

In this section, we show how we can take advantage of basic conflicts for query answering over inconsistent databases. By conflict-aware query answering, we mean a query evaluation framework that first, in a preprocessing step, detects the collection of basic conflicts in an input inconsistent database, and then takes advantage of this collection to do useful reasoning for answering queries without adding a significant overhead to the time and space required for usual query evaluation. To this end, we use annotated databases and introduce an annotated relational algebra, which simply adds a number of annotation propagation rules to the classical relational algebra semantics.

Let *Annt* be a set of annotations, e.g., the set of natural numbers, and *Dom* be the domain of all attributes. An *annotated database instance* $\hat{I}$ of a database schema $S = \{R_1, \ldots, R_l\}$ consists of a set of annotated relation instances $\hat{I} = \{\hat{I}_{R_1}, \ldots, \hat{I}_{R_l}\}$. Each annotated instance $\hat{I}_{R_i}$ of $R_i(A_1, \ldots, A_m)$ is a set of annotated tuples of the form $(\hat{t}, \mathcal{S}, \mathcal{D})$, where $\hat{t}$ is an m-tuple of (annotation, value) pairs, and $\mathcal{S} \subseteq Annt$, $\mathcal{D} \subseteq 2^{Annt}$ are used for recording provenance information for selection operators $\sigma_{A=a}$, $\sigma_{A=A'}$ respectively.

For an attribute A, $\hat{t}[A].val$ corresponds to the value of attribute A, and $\hat{t}[A].annt$ denotes the annotation. The underlying m-tuple of attribute values obtained by ignoring the annotations of $\hat{t}$ is denoted by $\hat{t}.val$, and the m-tuple of annotations appearing in $\hat{t}$ is denoted by $\hat{t}.annt$. The set of all such annotations in tuple $\hat{t}$ is referred to by $\hat{t}.aset$. For instance, for the first annotated tuple $(\hat{t}, \mathcal{S}, \mathcal{D})$ in Figure 4(b), we have $\hat{t}.val = (a_1, b_1, c_1)$, $\hat{t}.annt = (1, 2, 3)$, and $\hat{t}.aset = \{1, 2, 3\}$. Similarly, for a set of annotated tuples $\hat{S}$, $\hat{S}.val$ denotes the set of classical tuples $\{t \mid t = \hat{t}.val, (\hat{t}, \mathcal{S}, \mathcal{D}) \in \hat{S}\}$, and $\hat{S}.aset$ denotes the set of annotations appearing in $\hat{t}$, $\mathcal{S}$, or $\mathcal{D}$ for all annotated tuples $(\hat{t}, \mathcal{S}, \mathcal{D}) \in \hat{S}$.

Using these notations, in Figure 3, we define the semantics of a simple annotated relational algebra, which we need for propagating the conflicts to query answers. The purpose of this annotated algebra is first copying the annotations of the data values in the input relation to the values appearing in the answer, and then producing a light form of provenance by saving the annotations of values that have been involved in the selections. Intuitively, the provenance sets $\mathcal{S}, \mathcal{D}$ for each annotated tuple $(\hat{t}, \mathcal{S}, \mathcal{D})$ carry the annotation of data values that

$$\sigma_{A=a}(\hat{I}_R) = \{(\hat{t}, \mathcal{S}, \mathcal{D}) \mid (\hat{t}, \mathcal{S}', \mathcal{D}) \in \hat{I}_R,\ \hat{t}[A].val = a,\ \mathcal{S} = \mathcal{S}' \cup \{\hat{t}[A].annt\}\}$$
$$\sigma_{A=A'}(\hat{I}_R) = \{(\hat{t}, \mathcal{S}, \mathcal{D}) \mid (\hat{t}, \mathcal{S}, \mathcal{D}') \in \hat{I}_R,\ \hat{t}[A].val = \hat{t}[A'].val,\ \mathcal{D} = \mathcal{D}' \cup \{\{\hat{t}[A].annt, \hat{t}[A'].annt\}\}\}$$
$$\pi_X(\hat{I}_R) = \{(\hat{t}[X], \mathcal{S}, \mathcal{D}) \mid (\hat{t}, \mathcal{S}, \mathcal{D}) \in \hat{I}_R\}$$
$$\hat{I}_{R_1} \times \hat{I}_{R_2} = \{(\langle \hat{t}_1, \hat{t}_2 \rangle, \mathcal{S}, \mathcal{D}) \mid (\hat{t}_1, \mathcal{S}_1, \mathcal{D}_1) \in \hat{I}_{R_1},\ (\hat{t}_2, \mathcal{S}_2, \mathcal{D}_2) \in \hat{I}_{R_2},\ \mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2,\ \mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2\}$$
$$\hat{I}_R^1 \cup \hat{I}_R^2 = \{(\hat{t}, \mathcal{S}, \mathcal{D}) \mid (\hat{t}, \mathcal{S}, \mathcal{D}) \in \hat{I}_R^1 \text{ or } (\hat{t}, \mathcal{S}, \mathcal{D}) \in \hat{I}_R^2\}$$
$$\rho_{A' \leftarrow A}(\hat{I}_R) = \{(\hat{t}, \mathcal{S}, \mathcal{D}) \mid (\hat{t}, \mathcal{S}, \mathcal{D}) \in \hat{I}_R\}$$

Fig. 3. Annotated relational algebra

A	B	C
a_1	b_1	c_1
a_2	b_1	c_1

(a)

A	B	C	$\mathcal{S}$	$\mathcal{D}$
$1,a_1$	$2,b_1$	$3,c_1$	$\emptyset$	$\emptyset$
$4,a_2$	$5,b_1$	$6,c_1$	$\emptyset$	$\emptyset$

(b)

A	$\mathcal{S}$	$\mathcal{D}$
$1,a_1$	$\{2\}$	$\emptyset$
$4,a_2$	$\{5\}$	$\emptyset$

(c)

Fig. 4. (a) database instance **(b)** annotated instance **(c)** after running $\pi_A(\sigma_{B=b_1}(\hat{I}))$

have contributed to the selection of the tuple. A single annotation is added to $\mathcal{S}$ whenever the single selection operator $\sigma_{A=a}$ is performed, and a set of two annotations is added to $\mathcal{D}$ whenever the double selection operator $\sigma_{A=A'}$ is performed. For simplicity, we sometimes write $\mathcal{S} \cup \mathcal{D}$ to denote the set of annotations that appear in $\mathcal{S}$ or $\mathcal{D}$.

Given an instance I of a database schema $S = \{R_1, \ldots, R_l\}$, we create an annotated relational database $\hat{I}$ by giving a unique annotation $\hat{p} \in [1, |Pos(I)|]$ to each position $p \in Pos(I)$. That is, $Annt = \{1, \ldots, |Pos(I)|\}$. The annotated database instance $\hat{I}$ consists of $\hat{I}_{R_1}, \ldots, \hat{I}_{R_l}$, where

$$\hat{I}_{R_i} = \{(\hat{t}, \emptyset, \emptyset) \mid \hat{t} = ((\hat{p}_1, I(p_1)), \ldots, (\hat{p}_m, I(p_m))),\ \text{where}\ (I(p_1), \ldots, I(p_m)) \in I_{R_i}\}.$$

Intuitively, $\hat{I}$ annotates every position p in the instance with a unique number $\hat{p}$ and initializes the provenance sets to be empty (see Figure 4(b)). If I is an inconsistent database, then each minimal repair I' of I can have a similar representation $\hat{I}'$: for each annotated tuple $(\hat{t}, \emptyset, \emptyset)$ in $\hat{I}$, there is $(\hat{t}', \emptyset, \emptyset)$ in $\hat{I}'$ with $\hat{t}.annt = \hat{t}'.annt$, but the values in $\hat{t}.val$ and $\hat{t}'.val$ may not agree.

For a relational algebra query Q, we write $Q(I)$ to denote the classical answer obtained by usual evaluation of Q over I, and $\hat{Q}(\hat{I})$ to denote the annotated answer obtained by the rules shown in Figure 3. Similarly, we use t (resp., $(\hat{t}, \mathcal{S}, \mathcal{D})$) to denote a classical (resp., annotated) tuple. For a tuple $t \in Q(I)$, an annotated tuple $(\hat{t}, \mathcal{S}, \mathcal{D}) \in \hat{Q}(\hat{I})$ is called a *derivation* of t if $\hat{t}.val = t$. Similarly, for a set of tuples $S \subseteq Q(I)$, a set of annotated tuples $\hat{S} \subseteq \hat{Q}(\hat{I})$ is called a derivation of S if $\hat{S}.val = S$. Intuitively, a derivation is one of the possibly many ways of obtaining a (set of) tuple(s) in the answer.

3.1 Answers with Consistent Derivation

Let I be an inconsistent database instance that violates a set of (conditional) functional dependencies Σ. Now we define *answers with consistent derivation* for a relational algebra query Q.

Definition 2. *A set of tuples $S \subseteq Q(I)$ has a* consistent derivation *if S has a derivation $\hat{S} \subseteq \hat{Q}(\hat{I})$ such that $\hat{S} \subseteq \hat{Q}(\hat{I}')$ for some minimal repair I'.*

Intuitively, S has a consistent derivation if for some minimal repair I', all the tuples in S appear in $Q(I')$ through the same set of positions. That is, one of the derivations of S persists in $\hat{Q}(\hat{I}')$. Notice that this definition does not suggest that answers with consistent derivation could be efficiently computed, because the number of minimal repairs could potentially be exponential in the size of inconsistent database. Suppose that the collection of basic conflicts Σ-*Confs*(I) is given, where each basic conflict is represented as a subset of $[1, |Pos(I)|]$. Next we would like to show that after having Σ-*Confs*(I), we can capture answers with consistent derivation at no extra computational cost for query evaluation.

Example 3. Figure 4(b) shows the annotated version $\hat{I}$ of database instance I in Figure 4(a). The result of running the query $\pi_A(\sigma_{B=b_1}(\hat{I}))$ is shown in Figure 4(c). Consider the set of CFDs $\Sigma = \{(A \to C, (a_1, c_1)), (B \to C, (b_1, c_2)),$ stating that if A is a_1, then C should be c_1, and if B is b_1, then C should be c_2. The set of basic conflicts for instance I is Σ-*Confs*$(I) = \{\{1,2\},\{2,3\},\{5,6\}\}$. Notice that the subset $\{(a_1)\}$ of the answer does not have a consistent derivation, and this is reflected in the first annotated tuple of Figure 4(c): the annotations in this tuple contain a basic conflict. $\square$

The next theorem shows that $S \subseteq Q(I)$ has a consistent derivation if it has a derivation $\hat{S} \subseteq \hat{Q}(\hat{I})$ that contains none of the basic conflicts in Σ-*Confs*(I), i.e., if S could be obtained by a non-conflicting set of values.

Theorem 2. (Soundness) *Let Q be a positive relational algebra query. A set of tuples $S \subseteq Q(I)$ has a consistent derivation if S has a derivation $\hat{S} \subseteq \hat{Q}(\hat{I})$, such that $\hat{S}.aset$ does not contain any of the basic conflicts in Σ-Confs(I).*

We achieve completeness for queries that satisfy either of these restrictions:

Restriction I: Query has no projection.
Restriction II: For every attribute A in one of the base relations, there is at most one selection operator $\sigma_{A=a}$ in the query. Furthermore, for every selection operator $\sigma_{A=A'}$, either A or A' refer to a clean attribute.

Theorem 3. (Completeness) *Let Q be a positive relational algebra query that satisfies* Restriction I *or* Restriction II*. A set of tuples $S \subseteq Q(I)$ has a consistent derivation only if S has a derivation $\hat{S} \subseteq \hat{Q}(\hat{I})$, such that $\hat{S}.aset$ does not contain any of the basic conflicts in Σ-Confs(I).*

Remark. We can remove the first condition for Restriction II if we have more sets like $\mathcal{S}$ in the annotated tuples to record the provenance of all selection operators of the form $\sigma_{A=a}$.

3.2 Possible Answers

In the previous section, we showed how we can efficiently check whether a set of tuples in the answer of a query over an inconsistent database can be obtained by running the query over some minimal repair with the same derivation. However, we may be interested in answers that can be obtained from some minimal repair regardless of the derivation. This is referred to as *possible answer checking*, which is a well-studied problem for incomplete databases (see [1]). Possible answers for inconsistent databases can be defined as follows.

Let I be an inconsistent database instance that violates a set of (conditional) functional dependencies Σ, and Q be a relational algebra query.

A	B	$\mathcal{S}$	$\mathcal{D}$
1,a_1	2,b_1	$\emptyset$	$\emptyset$
3,a_2	4,b_2	$\emptyset$	$\emptyset$

(a)

A	B	$\mathcal{S}$	$\mathcal{D}$
1,a_1	2,b_1	$\emptyset$	$\emptyset$
1,a_1	4,b_2	$\emptyset$	$\emptyset$
3,a_2	2,b_1	$\emptyset$	$\emptyset$
3,a_2	4,b_2	$\emptyset$	$\emptyset$

(b)

Fig. 5. (a) annotated instance **(b)** after running $\pi_A(\hat{I}) \times \pi_B(\hat{I})$

Definition 3. *A set of tuples $S \subseteq Q(I)$ is a* possible answer *if $S \subseteq Q(I')$ for some minimal repair I'.*

In this paper, we are interested in possible answers with bounded size, $|S|$, because it is easy to show that if the size of S is not bounded, then possible answer checking becomes NP-complete for very simple queries. This is not a surprising result as unbounded possible answer checking is NP-complete for some positive queries over simple representations of incomplete databases [1].

Proposition 1. *There is a set of functional dependencies Σ with only one FD and a query Q with natural join and projection ($\pi, \sigma_{A=A'}, \times$), for which unbounded possible answer checking is NP-complete.*

Next, we would like to show how conflict-aware query answering can help us identify possible answers of bounded size. Soundness of this framework is an immediate corollary of soundness for answers with consistent derivation.

Corollary 1. (Soundness) *Let Q be a positive relational algebra query. A set of tuples $S \subseteq Q(I)$ is a possible answer if S has a derivation $\hat{S} \subseteq \hat{Q}(\hat{I})$, such that $\hat{S}.aset$ does not contain any of the basic conflicts in Σ-Confs(I).*

For some queries, however, a set of tuples can be a possible answer without having a consistent derivation in the inconsistent database. In other words, a minimal repair could make changes to the inconsistent database in a way that those tuples could be obtained in the answer through a different set of positions, and realizing this may require additional reasoning.

Example 4. Consider CFDs $\Sigma = \{(A \rightarrow B, (a_1, b_2)), (A \rightarrow B, (a_2, b_1))\}$. For the instance I shown in Figure 5(a), we have Σ-*Confs*$(I) = \{\{1, 2\}, \{3, 4\}\}$. Observe that for the query $Q(I) = \pi_A(I) \times \pi_B(I)$, the tuple (a_1, b_1) is a possible answer, while it does not have a consistent derivation in $\hat{Q}(\hat{I})$ in Figure 5(b). □

Nonetheless, there are still some queries for which we can efficiently check whether a set of tuples is a possible answer using our light-weight conflict-aware query evaluation framework. More specifically, we achieve completeness for queries satisfying any of the following restrictions, which implies that possible answers and answers with consistent derivation coincide for these queries. In these restrictions, by self product we mean the Cartesian product of two expressions that have a relation name in common.

RESTRICTION III: Query has no projection, no union, at most one self product.

Restriction IV: Query has no union and no self product. Furthermore, for every attribute A in one of the base relations, there is at most one selection operator $\sigma_{A=a}$ in the query. Also $\sigma_{A=A'}$ is allowed only when both A and A' refer to a clean attribute.

Theorem 4. (Completeness) *Let Q be a positive relational algebra query that satisfies* Restriction III *or* Restriction IV. *A set of tuples $S \subseteq Q(I)$ is a possible answer only if S has a derivation $\hat{S} \subseteq \hat{Q}(\hat{I})$, such that $\hat{S}.aset$ does not contain any of the basic conflicts in Σ-Confs(I).*

4 Generating Repairs Using Conflicts

One way to deal with inconsistency in databases is data cleaning: modifying attribute values to generate repairs. The quality of generated repairs is usually evaluated by a distance or cost measure that represents how far the repair is to the original inconsistent database. An *optimum repair* is a repair that minimizes the distance measure. Here we define such a measure and show how the notion of basic conflicts can help generate repairs whose distance to the input inconsistent database is reasonably close to the minimum distance.

Given a database instance I that violates a set of (C)FDs Σ, we define the distance between I and a repair I' to be $\Delta(I, I') = \sum_{(R,t,A) \in \mathit{Diff}(I,I')} w(t)$, where $w(t)$ is the cost of making any update to tuple t and represents the level of certainty or accuracy placed by the user who provided the fact . Intuitively, $\Delta(I, I')$ shows the total cost of modifying attribute values that are different in I and I'. In the absence of tuple weights, we can assume they are all equal to 1.

It has previously been shown that finding an optimum repair I_{Opt} that minimizes a distance measure such as $\Delta(I, I')$ is an NP-hard problem [9,11,20,25]. Furthermore, it is also NP-hard to find an approximate solution I' whose distance to I is within a constant factor of the minimum distance, i.e., $\Delta(I, I') \leq \alpha \cdot \Delta_{\min}$ (where the factor α does not depend on (C)FDs) [25].

It is, however, possible to produce repairs whose distance to I is within a factor α_Σ of $\Delta_{\min}$ in polynomial time, where α_Σ is a constant that depends on the set of dependencies Σ. In [25], we presented an approximation algorithm for producing such repairs that works in two steps: first, the *initial* conflicting sets of positions are detected by looking at the dependency violations, and a candidate set of positions for modification is found by applying an algorithm that approximates the minimum hitting set for the collection of the initial conflicts. In the second step, the value of some additional positions is changed as there still might be some dependency violations as a result of the changes made in the first step. In this section, we show that a second step would not be necessary if we have the collection of *all* basic conflicts Σ-*Confs*(I). Note that the initial conflicts used in the repair algorithm of [25] is basically a subset of Σ-*Confs*(I).

Here we assume that the cardinality of each basic conflict $T \in \Sigma$-*Confs*(I) is bounded by b_Σ. That is $|T| \leq b_\Sigma$ for some constant b_Σ that depends only on Σ. In the next section, we will present a sufficient condition on the set of

```
Algorithm ApproximateOptRepair
Input: Instance $I$, (C)FDs $\Sigma$, Basic conflicts $\Sigma$-$Confs(I)$.
Output: Repair $I'$.
for every position $p = (R, t, A) \in Pos(I)$ do
    assign $w(p) := w(t)$;
find an approximation $H$ for minimum hitting set of $\Sigma$-$Confs(I)$;
apply HittingSetRepair for $I$, $\Sigma$, and $H$;
return the output $I'$;
```

Fig. 6. Algorithm for finding an approximation to optimum repair

dependencies Σ that guarantees this bound. We show that we can have a repair algorithm that approximates optimum repair within a factor of b_Σ simply by applying a standard greedy algorithm that finds an approximate solution H for minimum hitting set of Σ-*Confs*(I) (see [17,30]), and then changing the value of positions that fall in H. This is what algorithm ApproximateOptRepair (Figure 6) does. Intuitively, since we resolve all the conflicts in the instance in one step, there is no need for a second step to take care of the new (C)FD violations caused by the modifications made in the first step. This highlights the importance of detecting all basic conflicts.

Theorem 5. *For every input inconsistent instance I,* ApproximateOptRepair *generates a repair I' with $\Delta(I, I') \leq b_\Sigma \cdot \Delta_{min}$, where b_Σ is a constant that depends only on Σ.*

5 Detecting Conflicts

In previous sections, we have seen how having the collection of basic conflicts can be helpful in both conflict-aware query answering and in finding repairs. A natural question is whether it is possible to find this collection efficiently. In particular, we are interested in the following question: whether for any inconsistent instance I of a given schema and set Σ of (conditional) functional dependencies, we can compute Σ-*Confs*(I) in polynomial time in the size of I. We show that the answer is yes if Σ does not have a *sink-free* cycle, defined below. In fact in that case, we show that there is a bound on the size of basic conflicts for any inconsistent instance. For simplicity, we do not discuss the case when Σ is a set of CFDs (when pattern tuples exist), but the results of this section can easily be applied to CFDs as well, simply by ignoring pattern tuples.

For a set of FDs Σ, we define a directed graph $\mathcal{G}_\Sigma(\Sigma, E)$, where Σ is the set of vertices, and E is the set of edges, such that $(X_1 \to A_1, X_2 \to A_2) \in E$ whenever $A_1 \in X_2$. Let $\mathcal{C} = \{X_1 \to A_1, \ldots, X_k \to A_k\}$ be a subset of Σ that forms a cycle in $\mathcal{G}_\Sigma$. That is, $A_i \in X_{i+1}$ for $i \in [1, k)$ and $A_k \in X_1$. Then $\mathcal{C}$ is called *sink-free* if for every FD $X_i \to A_i \in \mathcal{C}$, there exists an FD $X_j \to A_j \in \mathcal{C}$, such that $\Sigma \not\models X_j \to X_i$. The following example shows how an instance that violates a set of FDs with a sink-free cycle can have arbitrarily large basic conflicts.

Example 5. Consider schema $R(A, B, C, D)$ with FDs $\Sigma = \{AB \to C, CD \to A\}$. The FDs in Σ form a sink-free cycle since $\Sigma \not\models AB \to CD$ and $\Sigma \not\models$

A	B	C	D
a_1	b_1	c_1	
a_1	b_1		d_1
a_2		c_1	d_1

A	B	C	D
a_1	b_1	c_1	
a_1	b_1		d_1
	b_2	c_1	d_1
a_1	b_2		d_2
a_2		c_1	d_2

A	B	C	D
a_1	b_1	c_1	
a_1	b_1		d_1
	b_2	c_1	d_1
a_1	b_2		d_2
	b_3	c_1	d_2
a_1	b_3		d_3
a_2		c_1	d_3

Fig. 7. Instances with large basic conflicts

```
**Algorithm** ChaseExpansion
**Input:** Instance $I$, FDs $\Sigma$, Positions $T \subseteq Pos(I)$.
**Output:** Chase expansion of $T$ w.r.t. $\Sigma$.
$T^+ := T$;
**while** there is an FD $X \to A \in \Sigma$ and tuples $t_1, t_2$ in $I_R$ such that $t_1[X] = t_2[X]$ and
$\{(R, t_i, B) \mid i \in [1, 2], B \in XA\} \setminus T^+ = \{(R, t_2, A)\}$ **do**
    $I(R, t_2, A) := I(R, t_1, A)$;
    $T^+ := T^+ \cup \{(R, t_2, A)\}$;
**return** $T^+$;
```

Fig. 8. Chase expansion of a set of positions w.r.t. a set of FDs

$CD \to AB$. Figure 7 shows three instances of R, where the filled positions form a basic conflict. Observe that we can create instances with arbitrarily-large basic conflicts by repeating the pattern that exists in these instances. □

Next, we show that inconsistent instances can have arbitrarily-large basic conflicts only if the set of FDs has a sink-free cycle. First, we define the notions of *chase expansion* and *chase DAG* (directed acyclic graph) for a set of positions in the database. Let $T \subseteq Pos(I)$. The chase expansion of T, denoted by T^+, is the output of the chase procedure shown in Figure 8. Each step of this procedure adds one position (R, t_2, A) to T^+ if there is an FD $X \to A$ and two tuples t_1, t_2 that agree on X, of which the only position not already in T^+ is (R, t_2, A). The value in position (R, t_2, A) of I is also updated to the value in (R, t_1, A). Clearly, a set of positions $T \subseteq Pos(I)$ contains a basic conflict in Σ-*Confs*(I) if and only if the chase expansion of T w.r.t. Σ contains a violation of an FD in Σ^+.

Let $(t_1, t_2, X \to A)$ denote a step in the chase expansion of T, and N denote the set of all these steps. We define chase DAG of T to be the DAG $\mathcal{D}(N, F)$, with vertices N and edges F, where there is an edge from $(t_1, t_2, X_1 \to A_1)$ to $(t_2, t_3, X_2 \to A_2)$ in F whenever $A_1 \in X_2$. Clearly, $(t_2, t_3, X_2 \to A_2)$ is a step that takes place after $(t_1, t_2, X_1 \to A_1)$ in the chase, and, intuitively, it shows that the value produced by the former step is being used by the latter by being placed on the left-hand side of an FD application.

Theorem 6. *For every set of FDs Σ that does not have a sink-free cycle, there exists a number b_Σ, such that for every inconsistent instance I and every basic conflict $T \in \Sigma$-Confs(I), we have $|T| \leq b_\Sigma$.*

Proof sketch. Suppose there is a set of FDs Σ, such that for every integer l, there is an inconsistent instance I and a basic conflict $T \in \Sigma$-*Confs*(I) with $|T| > l$. We need to show that Σ has a sink-free cycle. Note that if for a set of FDs, the depth of chase DAG (length of the longest path from a leaf to a root) for every basic conflict is bounded, then an arbitrarily large basic conflict cannot exist, because the in-degree of vertices is bounded. We therefore assume that for such a set of FDs and every integer l, there is an instance I, such that for a basic conflict $T \in \Sigma\text{-}Confs(I)$, the depth of the chase DAG of T is larger than l.

We can assume, w.l.o.g., that the chase DAG of T is a chain, meaning that each vertex has at most one incoming and one outgoing edge. This is because for every instance I, a basic conflict $T \in \Sigma\text{-}Confs(I)$, and a path q from a leaf to a root in the chase DAG of T, we can create another instance I' with a basic conflict $T' \in \Sigma\text{-}Confs(I')$ whose chase DAG is a chain that exactly looks like q (it is enough to make the following updates to I: for every chase step $(t_1, t_2, X \rightarrow A)$ not on q, we change the value of $t_2[A]$ in I to the value that the chase step introduces). Next, observe that the depth of any chase DAG is bounded by the depth of the FD graph $\mathcal{G}_\Sigma(\Sigma, E)$ if Σ does not have a cycle. Thus, Σ should be cyclic. Suppose cyclic FDs $\mathcal{C} = \{X_1 \rightarrow A_1, \ldots, X_k \rightarrow A_k\} \subseteq \Sigma$ are repeatedly applied in the chain: $X_{i+1} \rightarrow A_{i+1}$ is applied right after $X_i \rightarrow A_i$ for $i \in [1, k)$ $(A_i \in X_{i+1})$, and $X_1 \rightarrow A_1$ is applied right after $X_k \rightarrow A_k$ $(A_k \in X_1)$ and so on. Next, we observe that if an FD, e.g., $X_1 \rightarrow A_1$, is applied twice in a chain for two different chase steps $(t_1, t_2, X_1 \rightarrow A_1)$ and $(t_{k+1}, t_{k+2}, X_1 \rightarrow A_1)$, then $t_1[X_1] \neq t_{k+1}[X_1]$ in the chase expansion of T. Otherwise, we can directly perform the chase step $(t_1, t_{k+2}, X_1 \rightarrow A_1)$, and the path between these two steps is not necessary. This would contradict with the fact that T is a minimal set. Note that right after the chase step $(t_{k+1}, t_{k+2}, X_1 \rightarrow A_1)$ is performed, T^+, the partial chase expansion until this step, would not have contained an FD violation. Otherwise, the chase would have reached a violation before visiting all the positions in T, which implies that T has a proper subset that is a basic conflict. It is easy to observe that $t_1[X_1]$ can be different from $t_{k+1}[X_1]$ without T^+ containing a violation only if X_1 is not implied by all the left-hand sides of the FDs applied between the two chase steps. That is, there must be $X_j \rightarrow A_j \in \mathcal{C}$ such that $\Sigma \not\models X_j \rightarrow X_1$, which means Σ has a sink-free cycle. □

The main result of this section can be stated as a corollary to Theorem 6.

Corollary 2. *For every set of FDs Σ that does not have a sink-free cycle, and every inconsistent database instance I, the set of basic conflicts $\Sigma\text{-}Confs(I)$ can be computed in polynomial time in the size of I.*

6 Conclusions

We presented the notion of basic conflicts in inconsistent databases that violate a set of (conditional) functional dependencies. We considered possible answers, and introduced answers with consistent derivation as a more restrictive notion of possibility. By annotating databases and propagating annotations during query evaluation, we showed how we can identify possible answers or answers with consistent derivation by checking whether the annotated answers contain a basic conflict. Then we showed that basic conflicts could also be used in data cleaning, when the goal is generating repairs that are close to the input inconsistent database according to a distance measure. We characterized dependencies for which the size of each basic conflict is bounded for any inconsistent database, and thus the collection of basic conflicts is computable in polynomial time.

Other problems for inconsistent databases could be reexamined using the notion of basic conflicts. For instance, it would be interesting to find out whether there is any connection between conflict-aware query answering and consistent query answering for finding certain answers. We would also like to examine conflict-aware query answering for more expressive queries, such as queries with inequalities. It would also be interesting to look for efficient algorithms for generating the collection of basic conflicts.

Acknowledgments. This work was supported by a grant from the Natural Sciences and Engineering Research Council of Canada for the Business Intelligence Network, which is gratefully acknowledged.

References

1. Abiteboul, S., Kanellakis, P.C., Grahne, G.: On the representation and querying of sets of possible worlds. Theor. Comput. Sci. 78(1), 158–187 (1991)
2. Arenas, M., Bertossi, L.E., Chomicki, J.: Consistent query answers in inconsistent databases. In: PODS, pp. 68–79 (1999)
3. Arenas, M., Bertossi, L.E., Chomicki, J.: Answer sets for consistent query answering in inconsistent databases. TPLP 3(4-5), 393–424 (2003)
4. Arenas, M., Bertossi, L.E., Chomicki, J., He, X., Raghavan, V., Spinrad, J.: Scalar aggregation in inconsistent databases. Theor. Comput. Sci. 3(296), 405–434 (2003)
5. Arenas, M., Bertossi, L.E., Kifer, M.: Applications of annotated predicate calculus to querying inconsistent databases. In: Computational Logic, pp. 926–941 (2000)
6. Barceló, P., Bertossi, L.E., Bravo, L.: Characterizing and computing semantically correct answers from databases with annotated logic and answer sets. In: Semantics in Databases, pp. 7–33 (2001)
7. Benjelloun, O., Sarma, A.D., Halevy, A.Y., Widom, J.: Uldbs: Databases with unvertainty and lineage. In: VLDB, pp. 953–964 (2006)
8. Bertossi, L.E.: Consistent query answering in databases. SIGMOD Record 35(2), 68–76 (2006)
9. Bertossi, L.E., Bravo, L., Franconi, E., Lopatenko, A.: The complexity and approximation of fixing numerical attributes in databases under integrity constraints. Inf. Syst. 33(4-5), 407–434 (2008)
10. Bohannon, P., Fan, W., Geerts, F., Jia, X., Kementsietsidis, A.: Conditional functional dependencies for data cleaning. In: ICDE, pp. 746–755 (2007)
11. Bohannon, P., Flaster, M., Fan, W., Rastogi, R.: A cost-based model and effective heuristic for repairing constraints by value modification. In: SIGMOD Conference, pp. 143–154 (2005)
12. Buneman, P., Khanna, S., Tan, W.C.: Why and where: A characterization of data provenance. In: ICDT, pp. 316–330 (2001)
13. Buneman, P., Khanna, S., Tan, W.C.: On propagation of deletions and annotations through views. In: PODS, pp. 150–158 (2002)

14. Chomicki, J.: Consistent query answering: Five easy pieces. In: ICDT, pp. 1–17 (2007)
15. Chomicki, J., Marcinkowski, J.: Minimal-change integrity maintenance using tuple deletions. Inf. Comput. 197(1-2), 90–121 (2005)
16. Cong, G., Fan, W., Geerts, F., Jia, X., Ma, S.: Improving data quality: Consistency and accuracy. In: VLDB, pp. 315–326 (2007)
17. Hochbaum, D.S.: Approximation Algorithms for NP-Hard Problems. PWS (1997)
18. Fan, W.: Dependencies revisited for improving data quality. In: PODS, pp. 159–170 (2008)
19. Fan, W., Geerts, F., Jia, X., Kementsietsidis, A.: Conditional functional dependencies for capturing data inconsistencies. ACM Trans. Database Syst. 33(2) (2008)
20. Flesca, S., Furfaro, F., Parisi, F.: Consistent query answers on numerical databases under aggregate constraints. In: DBPL, pp. 279–294 (2005)
21. Fuxman, A., Miller, R.J.: First-order query rewriting for inconsistent databases. J. Comput. Syst. Sci. 73(4), 610–635 (2007)
22. Geerts, F., Kementsietsidis, A., Milano, D.: Mondrian: Annotating and querying databases through colors and blocks. In: ICDE, p. 82 (2006)
23. Grahne, G., Mendelzon, A.O.: Tableau techniques for querying information sources through global schemas. In: ICDT, pp. 332–347 (1999)
24. Greco, S., Molinaro, C.: Approximate probabilistic query answering over inconsistent databases. In: Li, Q., Spaccapietra, S., Yu, E., Olivé, A. (eds.) ER 2008. LNCS, vol. 5231, pp. 311–325. Springer, Heidelberg (2008)
25. Kolahi, S., Lakshmanan, L.V.S.: On approximating optimum repairs for functional dependency violations. In: ICDT, pp. 53–62 (2009)
26. Kolahi, S., Lakshmanan, L.V.S.: Exploiting conflict structures in inconsistent databases (2010) (full version), http://www.cs.ubc.ca/~solmaz/conflict-proofs.pdf
27. Levene, M., Loizou, G.: Database design for incomplete relations. ACM Trans. Database Syst. 24(1), 80–125 (1999)
28. Levene, M., Loizou, G.: A Guided Tour of Relational Databases and Beyond. Springer, London (1999)
29. Lopatenko, A., Bravo, L.: Efficient approximation algorithms for repairing inconsistent databases. In: ICDE, pp. 216–225 (2007)
30. Vazirani, V.V.: Approximation Algorithms. Springer, Heidelberg (2003)
31. Wijsen, J.: Database repairing using updates. ACM Trans. Database Syst. 30(3), 722–768 (2005)
32. Wijsen, J.: Consistent query answering under primary keys: a characterization of tractable queries. In: ICDT, pp. 42–52 (2009)

Satisfiability and Containment Problem of Structural Recursions with Conditions

Balázs Kósa*, András Benczúr, and Attila Kiss

Eötvös Loránd University, Faculty of Informatics,
1118, Budapest, Pázmány Péter sétány 1/C
{balhal,abenczur,kiss}@inf.elte.hu

Abstract. Structural recursion is a general graph traversing and restructuring operation. In [9][10] we extended structural recursions with not-isempty conditions and a limited form of registers in order to simulate a large fragment of XPath 1.0 and XSLT 1.0. In this paper we address the satisfiability and containment problem of structural recursions with not-isempty conditions. We examine two cases, when else-branches are allowed in the conditions and when they are not. In the second case it turns out that the satisfiability problem can be solved in quadratic time, while the question of containment is coNP-complete in general. In the first case both problems are PSPACE-complete in general. Our results have their theoretical importance of their own, however, we hope that they can also be used to fill in some of the gaps in the research of XPath and XSLT.

Keywords: Semistructured Data, XML, Structural Recursion.

1 Introduction

Structural recursion is a graph traversing and restructuring operation applied on many fields of computer science including syntax analysis, code generation and program transformation. In the context of databases it was already recommended as a query language alternative in the early 90's to be able to overstep the limitations of the relational data model, and to move toward to the expressive capability of a relational query language embedded into a general purpose programming language [4]. Its expressive power suffice for most practical problems and its easy computation makes it a good candidate for this purpose.

The rising of semistructured databases and XML put structural recursions again in the limelight. It has formed the basis of UnQL [5], a query language for semistructured databases and also of XSLT. Several papers appeared then dealing with their properties [6][14]. However, later, other formal models were proposed [3][13], which superseded structural recursions.

* The research was supported by the project TÁMOP-4.2.1/B-09/1/KMR-2010-003 of Eötvös Loránd University.

In [8] we examined static analytical and optimization questions of simple structural recursions, i.e., structural recursions without conditions, in the presence of schema graphs [6]. In [9] we extended structural recursions with not-isempty conditions and a limited form of registers in order to simulate a large fragment of XPath 1.0. In [10] we showed how this simulation can be extended to a fragment of XSLT 1.0.

In this paper we address the question of satisfiability and containment for structural recursions with not-isempty conditions. Our results have their theoretical importance of their own, however, we hope that they can also be used in the static analysis of XPath and XSLT (acknowledging that most of the problems have been already solved in this field [2][12]. We examine two cases, when the else-branches are allowed in the conditions and when they are not. In the second case it turns out that the satisfiability problem can be solved in quadratic time, while the question of containment is coNP-complete in general. In the first case both problems are PSPACE-complete in general.

2 Preliminaries

XML documents and semistructured data are usually modelled with rooted, labelled, directed trees or graphs [1]. XML documents are represented as node-labelled trees in general, nonetheless, in our paper it is more convenient to use edge-labelled trees instead. Both node-labelled and edge-labelled trees can be

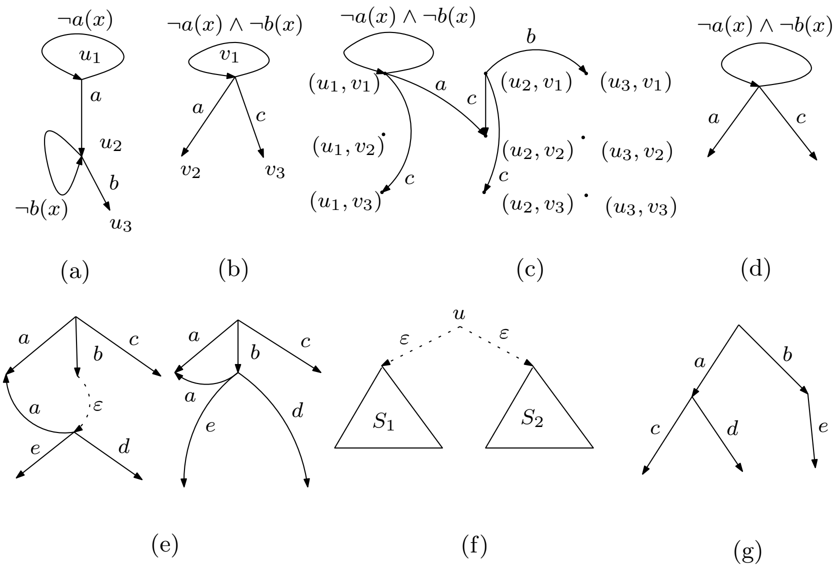


Fig. 1. (a) A data graph. (b) A data graph. (c) The intersection of the data graphs of (a) and (b). (d) The final result of this intersection. (e) An example of eliminating ε edges. (f) The union of data graphs. (g) $\{a : \{c : \{\}\} \cup \{d : \{\}\}\} \cup \{b : \{e : \{\}\}\}$.

transformed into each other without any difficulties [1]. We shall not enter into the details of the XML data model owing to our more theoretical approach, but it is not difficult to reformulate the followings in that setting.

Data graphs. Formally, let $\mathcal{U}$ be the universe of all constants ($\mathcal{U} = Int \cup String \cup \ldots$). Then a data graph (instance) I is a triplet $I = (V, E, v_0)$, where V is the set of nodes, $E \subset V \times \mathcal{U} \times V$ is the set of labelled edges and v_0 is the distinguished root [5]. $V.I$, $E.I$ denotes the sets of nodes and edges of an instance I respectively.

Basic notions of graphs. A sequence of subsequent edges of a graph, $a_1 \ldots a_n$ in notation, is called a *path*. A path, whose nodes are mutually different from each other, is said to be a *simple path*. By a *pregraph* we mean a subgraph of an instance I with the same root as that of I. Two edges having the same starting node are said to be *neighbours*. By a *cycle* we mean a simple path whose first and last node are the same. Looping edges will also be considered as cycles.

ε **edges.** ε edges will be used to ease the explanation. Their role is similar to the role of silent transitions in automata theory. At the end of computations they should be always eliminated. The elimination is accomplished in the following way: in an arbitrary instance I let $(u, \varepsilon, v) \in E.I$ be an ε edge. For every edge $(v, a, w) \in E.I$, where the starting node is the same as the end node of (u, ε, v), add (u, a, w) to $E.I$. With that delete the preceding (v, a, w) edges. Finally (u, ε, v) should also be deleted. As an example consider Figure 1.(e).

Union. For the union of arbitrary instances I, I' [5], let u be a new node different from all of the nodes of I and I'. Add ε edges from u to the root of I and I' and then eliminate these ε edges. For a graphical representation consider Figure 1.(f).

Data trees. Data trees can be built up using three constructors: the empty tree $\{\}$ consisting of a node only, the singleton set $\{l : t\}$, which is a directed l edge with subtree t in its end node, and the union $\cup$ operation. For example

$$\{a : \{c : \{\}\} \cup \{d : \{\}\}\} \cup \{b : \{e : \{\}\}\}$$

stands for the tree of Figure 1.(g). This construction also gives us a notation to represent data trees. These representations are said to be *ssd-expressions* [1] (ssd: semistructured data). In what follows $\{l_1 : t_1, \ldots, l_n : t_n\}$ will abbreviate $\{l_1 : t_1\} \cup \ldots \cup \{l_n : t_n\}$.

Operational graphs. *Operational graphs* [8] are rooted, directed graphs, whose edges are labelled with labels of the form $[p, a]$. Here p is a unary formula consisting of the Boolean combination of predicates $b(x)$ ($b \in \mathcal{U}$) and a is a constant in $\mathcal{U}$. We fix an interpretation throughout the paper, in which $b(x)$ becomes true iff $x = b$. A typical formula of this kind will look like as follows: $\neg b(x) \wedge \neg c(x)$. Note that the evaluation of p over this fixed interpretation can be accomplished in linear time. A $[p, a]$ labelled edge will represent that for those edges of data graphs, whose edge label satisfies p, an a edge should be constructed in the output.

Problems of satisfiability and containment. A structural recursion f is satisfiable, if there exists a data graph I s.t. $f(I)$ is not empty. For structural recursion f', f *contains* f', in notation $f \sqsupset f'$, if for all instances I, if I satisfies f, then I also satisfies f'.

3 Simple Structural Recursions

3.1 Tree Data Graphs

In order to describe the main features of structural recursions first we consider only trees as inputs. A structural recursion f is constituted by *structural functions*, in notation $f = (f_1, \ldots, f_n)$, which may call each other. The syntax rules of a structural function can be found in Figure 2. Note that here for a tree input t, we consider t as how it is built by constructors. In the definition we give that what should happen for the different constructors. Here, if f_i contains rows for singleton sets $\{a_1 : t\}, \ldots, \{a_k : t\}$, then $l_i = \neg a_1 \wedge \ldots \wedge \neg a_k$. If there are not such rows before the default case, then $l_i = \top$, where $\top$ denotes the predicate satisfied by all constants of $\mathcal{U}$. On the right hand side $\{l_i : \{\}\}$ means that an edge is constructed with the same label as that of the singleton set which is being processed.

We may have allowed the use of the union constructor on the right hand side, however, for the sake of transparency we have omitted this possibility. Our results can be extended to this general case without any serious changes.

As an example consider the following example $f = (f_1, f_2)$, which copies the subgraphs under *Ann* edges.:

$$\begin{array}{llll}
f_1 : (t_1 \cup t_2) = f_1(t_1) \cup f_1(t_2) & \qquad & f_2 : (t_1 \cup t_2) = f_2(t_1) \cup f_2(t_2) \\
f_1 : (\{Ann : t\}) = \{Ann : f_2(t)\} & & f_2 : (\{l_2 : t\}) = \{l_2 : f_2(t)\} \\
f_1 : (\{l_1 : t\}) = f_1(t) & & f_2 : (\{\}) = \{\} \\
f_1 : (\{\}) = \{\}. & &
\end{array}$$

Since the $f(t_1 \cup t_2)$, $\{\}$ rows are always the same, we shall not write them down in the the rest of the paper. Furthermore, we shall consider "rooted" structural recursions, which means that in all cases the first structural function f_1 is called on the root of the input.

a row of f_i	::=	$(t_1 \cup t_2) = f_i(t_1) \cup f_i(t_2) \mid (\{\}) = \{\} \mid$ $(\{a : t\}) = \text{singl\_set} \mid (\{l_i : t\}) = \text{singl\_set\_def}$
singl_set	::=	$\{\} \mid \{b : \{\}\} \mid f_j(t) \mid \{b : f_j(t)\}$
singl_set_def	::=	$\{\} \mid \{b : \{\}\} \mid \{l_i : \{\}\} \mid f_j(t) \mid \{b : f_j(t)\} \mid \{l_i : f_j(t)\}$

Fig. 2. The syntax of structural function f_i. Here the constants a, b are not necessarily different from each other. Similarly, f_i and f_j may also be the same structural functions.

3.2 Semantics

From now on we do not assume that the inputs are data trees. We define the semantics by means of intersection of operational graphs and data graphs [8].

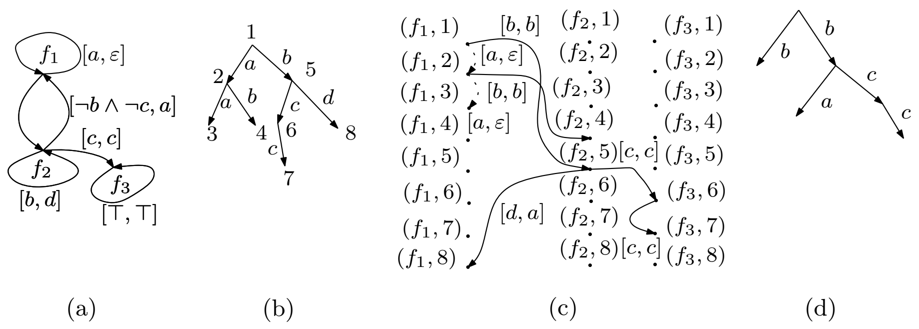


Fig. 3. (a) The operational graph of $f = (f_1, f_2, f_3)$. (b) A tree input, whose nodes are labelled with numbers for the sake of transparency. (c) The intersection of the operational graph of (a) and the tree input of (b). (d) The result.

First for a structural recursion $f = (f_1, \dots, f_n)$ we construct the corresponding operational graph, U_f in notation.

$|V.U_f| := n+1$, and let the names of the nodes be $f_1, \dots, f_n$ and w respectively. $E.U_f$ is given with respect to rows of f_i-s.

- For an $(\{a : t\}) = \{b : f_j(t)\}$ row (see Figure 2.), we add an $[a, b]$-labelled edge from f_i to f_j. Clearly, this edge represents that as a result of an a edge f_i calls f_j and constructs a b edge. If f_i is the same as f_j, a looping edge is constructed.
- We do not give in detail how for other types of rows the corresponding edge e in the operational graph should be constructed. We only note that in an $[a, \varepsilon]$ edge ε indicates that no edge is to be constructed in the result. If f_i does not call any structural functions for singleton set $(\{a : t\})$, e ends in w. Finally, an $[l_i, l_i]$ label shows that the label of the edge to be constructed is the same as the label of the processed input edge.

An example the operational graph of structural recursion $f = (f_1, f_2, f_3)$ can be seen in Figure 3.(a).

$$
\begin{array}{llll}
f_1: & (\{a : t\}) = f_1(t) & f_2: & (\{b : t\}) = \{d : f_2(t)\} \\
 & (\{l_1 : t\}) = \{b : f_2(t)\} & & (\{c : t\}) = \{c : f_3(t)\} \\
 & & & (\{l_2 : t\}) = \{a : f_1(t)\} \\
f_3: & (\{l_3 : t\}) = \{l_3 : f_3(t)\} & &
\end{array}
$$

Definition 1. *Let U_f be an operation graph and I a data graph. Then the intersection of U_f and I, $U_f \sqcap I$ in notation, is defined as follows: $V.U_f \sqcap I := \{(u, v) \mid u \in V.U_f, v \in V.I\}$, $E.U_f \sqcap I := \{((f_i, u), [a, \phi], (f_j, v)) \mid (f_i, [p, \phi], v_j) \in E.U_f, (u, a, v) \in E.I, \mathcal{U} \models p(a)\}, \phi \in \mathcal{U} \cup \{\varepsilon\}$.*

Note that edge $((f_i, u), [a, \phi], (f_j, v)) \in E.U_f \sqcap I$ shows that structural function f_i is called on edge $e = (u, a, v)$, a ϕ edge is constructed and f_j is called on the subtree under e. Examples can be found in Figure 3.(a)-(c). Now, in order to construct the result the second elements of the labels should be considered, and the ε edges should be eliminated. An example can be seen in Figure 3.(b)-(d).

4 Structural Recursions with Conditions

Again we begin with the syntax rules. These can be found in Figure 4. Here *n.i.* stands for the *not-isempty* function, whose domain is the set of data graphs, and it returns *true* iff its actual parameter is not the empty graph. Note that in the if-then-else conditions the else-branch is not required to give. What is more, we shall not consider embedded conditions. As an example consider the following structural recursion, which copies the subgraphs under b edges, if they contain an Ann edge:

$$\begin{aligned} f_1\colon (\{b:t\}) &= \text{if n.i.}(f_2(t)) \text{ then } \{b\colon f_3(t)\} \\ &\qquad\qquad\qquad\;\; \text{else } f_1(t) \\ (\{l_1:t\}) &= f_1(t) \end{aligned} \qquad \begin{aligned} f_2\colon (\{Ann\colon t\}) &= \{\psi:\{\}\} \\ (\{l_2\colon t\}) &= f_2(t) \\ f_3\colon (\{l_3\colon t\}) &= \{l_3\colon f_3(t)\} \end{aligned}$$

Definition 2. *Owing to their importance we give a distinguished name for structural recursions without else-branches. They will be called* pure structural recursions *in the sequel.*

a row of f_i	::=	$(t_1 \cup t_2) = f_i(t_1) \cup f_i(t_2) \mid (\{\}) = \{\} \mid$ $(\{a:t\}) =$ c_singl_s $\mid (\{l_i:t\}) =$ c_singl_s_def

c_singl_s	::=	$\{\} \mid \{b:\{\}\} \mid f_j(t) \mid \{b:f_j(t)\} \mid$
		 if n.i.$(f_j(t))$ then singl_set
		 if n.i.$(f_j(t))$ then singl_set else singl_set

c_singl_s_def	::=	$\{\} \mid \{b:\{\}\} \mid \{l_i:\{\}\} \mid f_j(t) \mid \{b:f_j(t)\} \mid \{l_i:f_j(t)\}$
		 if n.i.$(f_j(t))$ then singl_set_def
		 if n.i.$(f_j(t))$ then singl_set_def else singl_set_def

singl_set	::=	$\{\} \mid \{b:\{\}\} \mid f_j(t) \mid \{b:f_j(t)\}$
singl_set_def	::=	$\{\} \mid \{b:\{\}\} \mid \{l_i:\{\}\} \mid f_j(t) \mid \{b:f_j(t)\} \mid \{l_i:f_j(t)\}$

Fig. 4. The syntax of structural function f_i with not-isempty subqueries. Here the constants a, b are not necessarily different from each other. Similarly, f_i and f_j may also be the same structural functions.

4.1 Operational Graphs

As it is expected, the former definition is extended to this case. For a structural recursion $f = (f_1, \ldots, f_n)$, $|V.U_f| := 2n + 2$, and let the names of the nodes be $f_1, \ldots, f_n, f'_1, \ldots, f'_n, w, w'$ respectively.

The f'_i nodes will represent that case, when a structural function is called inside the check of a not-isempty condition. In this case the only important thing is that whether a construction is done or not. Thus an edge representing a construction will be linked to w' showing that the check of the non-emptiness of the result stops there. The subgraphs constituted by the f'_i, w' nodes and the edges between these nodes will be referred as the *conditional part*, while the pregraph of f_j, w nodes and the edges between these nodes will be called the *main part*. $E.U_f$ is given with respect to the rows of f_i-s again ($1 \leq i \leq n$).

- For rows without a condition the main part is constructed exactly in the same way as the operational graphs of simple structural recursions.

  In the conditional part, as we have already indicated, for the $\{b : f_j(t)\}$ rows an $[a, b]$ ($[l_i, b]$)-labelled edge is added from f'_i to w' instead of f'_j. Otherwise, the construction works in the same way as for the main part.
- For rows with a condition *if n.i.*$(f_j(t))$ *then* ... *else* ..., a polyedge is constructed. First an edge with label $[a, \varepsilon]$ ($[l, \varepsilon]$) is added from f_i (f'_i) to f'_j. Such an edge will be called *premise* in the sequel. These edges, however not all of them, connect the main part to the conditional part. Edges representing the then- and else-branches are added according to the rules given for the rows without condition.

  For instance in case of *if n.i.*$(f_j(t))$ *then* $\{b : f_k(t)\}$ *else* $\{f_l(t)\}$, an $[a, b]$-labelled and an $[a, \varepsilon]$-labelled edge is added to f_k and f_l in the main part respectively, while in the conditional part the $[a, b]$-labelled edge is added to w' and $[a, \varepsilon]$-labelled to f'_l. These edges will be respectively referred as *then-, else-edges*. The premise, then- and else-edge will be sometimes referred as *conditional edges*. Together they form a polyedge, $(f_i, [a, (b, \varepsilon)], (f'_j, f_k, f_l))$, $(f'_i, [a, (b, \varepsilon)], (f'_j, w', f'_l))$ for the previous example. In the sequel, since the check of n.i. condition starts there f_i, f'_i will be called *beginning of a condition* or shortly *b.o.c.*

A *constructor edge* is an edge representing a construction. This means that the second element of its label is different from ε.

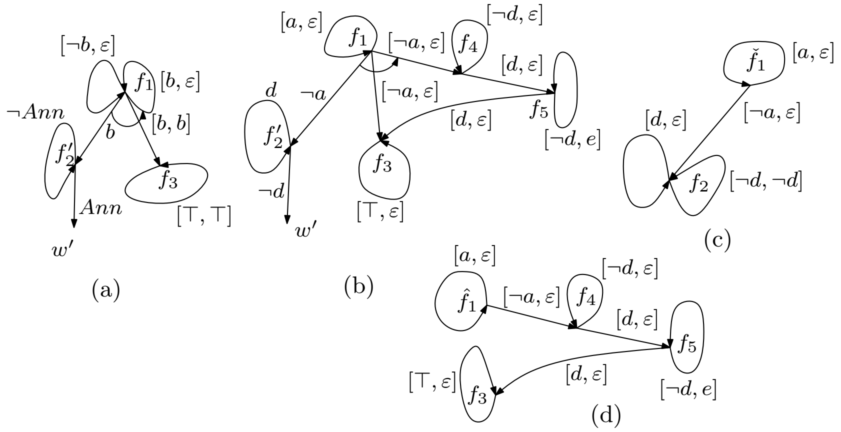


Fig. 5. (a) The operational graph of $f = (f_1, f_2, f_3)$. (b) The operational graph of Q_1. Here and in the rest of the paper the premise, then, else polyedge is drawn from left to right order. (c) $f_{Q_1}^{(f_1, \neg a, f'_2)}$. (d) $f_{Q_1}^{(f_1, [\neg a, \varepsilon], f_4)}$.

The operational graphs of structural recursions $f = (f_1, f_2, f_3)$ in the beginning of this section and of Q_1 on page 345 can be seen in Figure 5.(a)-(b). In Figure 5.(b) f_1 is a b.o.c., $(f_1, \neg a, f_2')$ is a premise, $(f_1, [\neg a, \varepsilon], f_3)$ is a then-, $(f_1, [a, \varepsilon], f_4)$ is an else-edge, while $(f_2', \neg d, w_1')$, $(f_5, [\neg d, e], f_5)$ are constructor edges.

Now, for each b.o.c. b find those constructor edges e that are reachable through a path not containing any premises except for the premise of b. Clearly, if a construction is done through e, then the appropriate condition belonging to b is satisfied. In the sequel, we shall say that *e belongs to b*. In Figure 5.(b) $(f_2', \neg d, u)$ belongs to f_1.

Note that the construction of an operational graph can be accomplished in time $O(n)$. Here n is the number of the rows of f. Furthermore from a given operational graph the original form of its structural recursion can be recovered without difficulties.

4.2 Definition of Semantics

Again we develop the semantics similarly as in the previous case. For a data graph I we construct $U_f \sqcap I$ and then consider only the output part of this intersection. However, in this case we should *reduce* the polyedges into normal edges first, i.e., for a polyedge we should keep only the then- or else-edge depending on the result of the not-isempty condition.

First, Definition 1. should be extended to the case, when an operational graph may contain polyedges. Owing to lack of space we only give an example of this straightforward extension in Figure 6.(a)-(c).

For an edge (e_{U_f}, e_I) in $E.U_f \sqcap I$, e_{U_f}, e_I are called *ancestor images* in U_f and I respectively. In $U_f \sqcap I$ e is a *premise* if its ancestor image in U_f is also a premise. The following notions: *then-, else-, constructor edge, b.o.c.* can be defined similarly in $U_f \sqcap I$. A constructor edge belongs to a premise in $U_f \sqcap I$ iff their ancestor images belongs together.

Condition evaluation. For the reduction of polyedges we develop an algorithm called *condition evaluation* that will decide whether a not-isempty condition is satisfied or not in $U_f \sqcap I$.

(i) In the first step consider those b.o.c.-s b from whose premise a constructor edge is reachable through a path not containing any conditional edge. Clearly, such a constructor edge e belongs to b. Obviously, if we applied the former semantics developed for simple structural recursions, then a construction would be done as a result of "reaching" e, i.e., the condition belonging to b would be satisfied. Hence we reduce the polyedge of b to its then-edge. In the following steps, the remaining then-edge edge is considered as a non-conditional edge. An example of this step can be seen in Figure 6.(a)-(d).

(ii) In case of those b.o.c.-s that do not have the former property, and from whose premise no other premise is reachable, we reduce their polyedges to the corresponding else-edges. Consider Figure 6.(d) for an example.

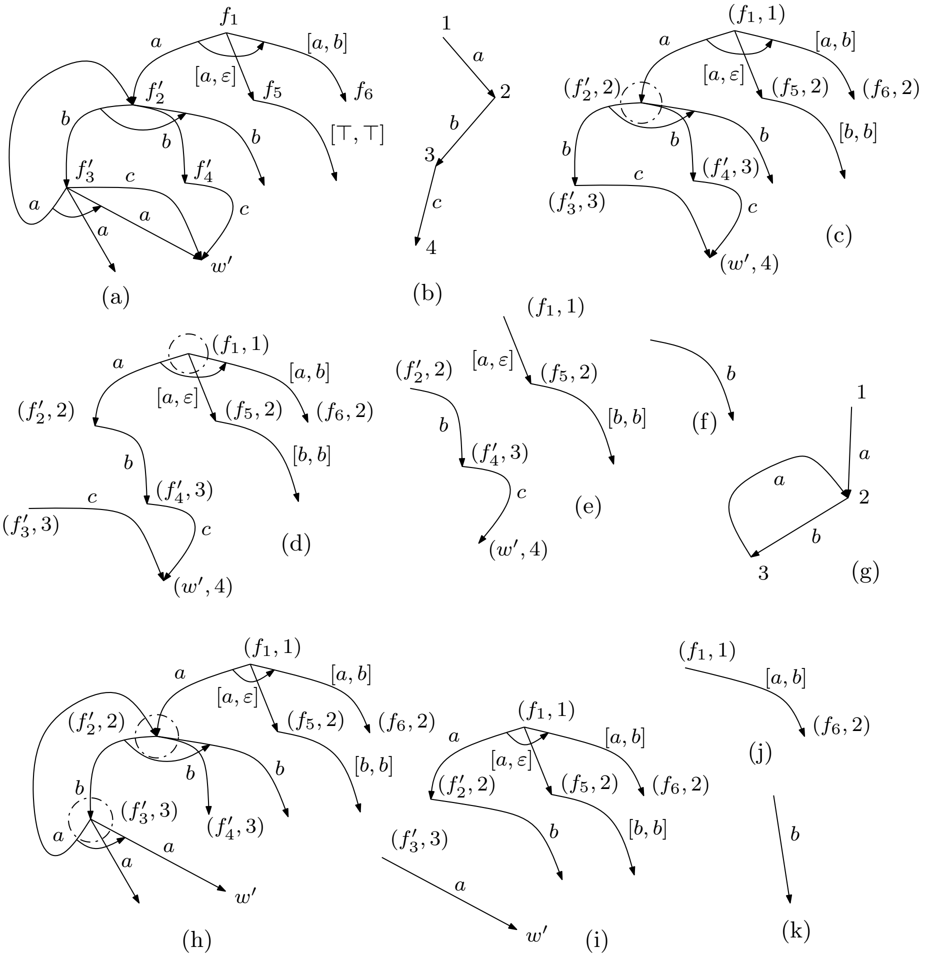


Fig. 6. (a) A part of an operational graph. (b) A tree input. (c) The intersection of the operational graph of (a) and the instance of (b). (d) The first step of the condition evaluation algorithm. The dashed-dotted circles indicate the reduced polyedges. (e) The second step. (f) The result. (g) An input with a cycle. (h) The intersection of the operational graph of (a) and the instance of (g). The dashed-dotted circles indicate b.o.c.-s whose premises form a cycle as it is described in the (iii) part of the condition evaluation algorithm. (i) The first step of the algorithm. (j) The second step. (k) The result.

In the next step we continue our algorithm with this new graph. Owing to the reduction of some polyedges, new polyedges may be reduced.

(iii) However, it may happen that steps (i), (ii) cannot be applied and there are still unreduced polyedges. It is easy to see that in this case some of the b.o.c.-s "form cycles" through their premise edges. Since the corresponding conditions cannot be satisfied their polyedges should be reduced to their else-edges. An example can be found in Figure 6.(h)-(j).

The algorithm stops, when all of the polyedges are reduced. Consider Figure 6.(a)-(e) for an example. Then $f(t)$ is defined as the output part of this graph with ε edges eliminated. See Figure 6.(f) and (k) for examples.

5 Polyedge Reduction and the Equivalence of the Problems of Satisfiability and Containment

Polyedge reduction. Note that since we have only used the notions of b.o.c., constructor edge and polyedge, which have been already defined for operational graphs, the former condition evaluation algorithm can also be applied to operational graphs instead of intersections. This form of the algorithm will be called *polyedge reduction*. As an example consider Figure 7.(a)-(d). Polyedge reduction gives us a sufficient condition for the unsatisfiability of a condition.

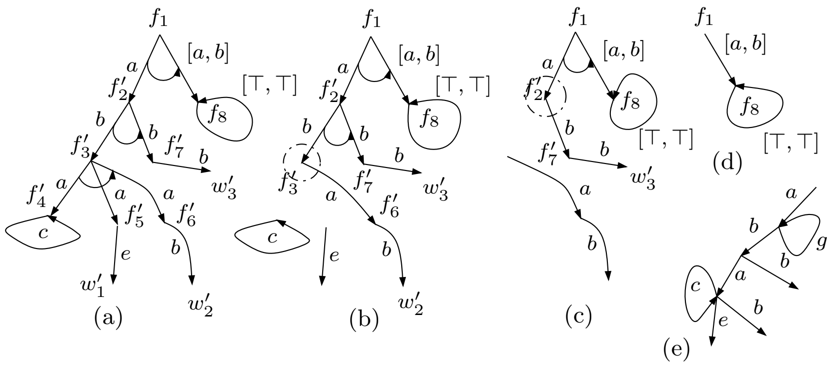


Fig. 7. (a) An operational graph.(b) The result of the first step of polyedge reduction applied on the operational graph of (a). (c) The second step. (d) The result.

Lemma 1. *For an arbitrary operational graph U_f, if in the polyedge reduction algorithm a polyedge is reduced to the else-edge, then the corresponding condition is not satisfiable.*

Equivalence of the problems of containment and satisfiability. To see that why this condition is not necessary, consider he following example.

Q_1: $f_1: (\{a : t\}) = f_1(t)$
$(\{l_1 : t\}) =$ if n.i.$(f_2(t))$ then $f_3(t)$ else $f_4(t)$

$f_2: (\{d : t\}) = f_2(t)$
$(\{l_2 : t\}) = \{l_2 : f_2(t)\}$

$f_3: (\{l_3 : t\}) = f_3(t)$

$f_4: (\{d : t\}) = f_5(t)$
$(\{l_4 : t\}) = f_4(t)$

$f_5: (\{d : t\}) = f_3(t)$
$(\{l_5 : t\}) = \{e : f_5(t)\}$

The corresponding operational graph can be seen in Figure 5.(b). Here those instances that may result in construction through the else-branch $\neg a.\neg d^*.d.\neg d$ of b.o.c. f_1, also "satisfy" the premise branch, $\neg a.d^*.\neg d$. Hence for such inputs the else branch is never called, consequently this structural recursion is unsatisfiable despite the fact that polyedge reduction reduces the polyedge of f_1 to its then-edge. Here d^* denotes arbitrary number of consecutive d edges.

To be able to examine this phenomenon in a general setting a new construction is needed. Let U_f be an operational graph and $po = (p, th, el)$ a polyedge, whose b.o.c. b is in the main part. Here p, th, el denote the premise, then- and else-edge of po respectively. Denote U_f^p (U_f^{th}, U_f^{el}) the subgraph of U_f reachable through p (th, el) with root b. The corresponding structural recursions of these subgraphs, f^p (f^{th}, f^{el}) will be called *premise (then, else) structural recursions belonging to p, (th, el)*. Consider $f_{Q_1}^{(f_1,\neg a,f_2')}$ and $f_{Q_1}^{(f_1,[\neg a,\varepsilon],f_4)}$ as examples in Figure 5.(c)-(d).

Clearly, Q_1 is unsatisfiable, for $f_{Q_1}^{(f_1,\neg a,f_2')}$ contains $f_{Q_1}^{(f_1,[\neg a,\varepsilon],f_4)}$. From this reasoning it follows that if there is an algorithm $\mathcal{A}$ solving the question of satisfiability for structural recursions in general, then the containment problem can also be decided by means $\mathcal{A}$. The reverse direction is also true. For deciding whether an arbitrary structural recursion f is satisfiable, it is enough to check whether $Q_1 \sqsupset f$ holds or not, where, remember, Q_1 is unsatisfiable. All in all we get the following theorem.

Theorem 1. *The question of satisfiability for structural recursions is in complexity class $\mathcal{C}$ in general iff the containment problem for structural recursions is in $\mathcal{C}$ in general.*

In the next section it will turn out that this theorem does not hold for pure structural recursions (Definition 2).

Note that one may ask the question whether there are conditions, for which it holds that any of the instances, which satisfy the structural recursion of the premise f^p, does not satisfy the structural recursion of the then-edge f^{th}. This would also mean that the corresponding condition is unsatisfiable. In the general case this may occur, however, it is not difficult to prove that in case of pure structural recursions, if instance I satisfies f^p and I' satisfies f^{th}, then $I \cup I'$ satisfies both f^p and f^{th}.

6 The Satisfiability and Containment Problem of Pure Structural Recursions

The satisfiability problem. In order to solve the satisfiability problem of pure structural recursion we use the "shape" of the operational graph. Namely, in a given operational graph U_f we pull the edges of a polyedge into one edge. Formally, from the end node of the premise we add ε edges to the end nodes of the then- and else-edge and then eliminate these ε edges. Then, in this new graph we substitute edge labels p ($[p, \theta]$) with constants a s.t. $\mathcal{U} \models p(a)$ ($\theta \in \{b, \varepsilon\}$). An example can be seen in Figure 7.(e). These instances will be called *matching instances of U_f*.

Lemma 2. *A given pure structural recursion is satisfiable iff a matching instance satisfies it.*

From the proof of this lemma it also follows that if a matching instance satisfies its structural recursion, then all of the matching instances satisfy it. Lemma 2 can be formulated in an other way.

Lemma 3. *A pure structural recursion is satisfiable iff after polyedge reduction a constructor edge is reachable from the root.*

Theorem 2. *The satisfiability problem of pure structural recursions can be solved in quadratic time.*

Problem of containment. We shall prove that the containment problem of pure structural recursions is coNP-complete. The question is clearly in coNP. To prove coNP-completeness we use the problem of deciding that whether a formula in CNF is a tautology or not, which is a well-known coNP-complete problem [7]. We explain the reduction by means of an example. First, we need a definition.

Definition 3. *For a structural recursion f, I is a minimal satisfier, if (i) $f(I)$ is non-empty, (ii) for each real pregraph I' of I $f(I')$ is empty.*

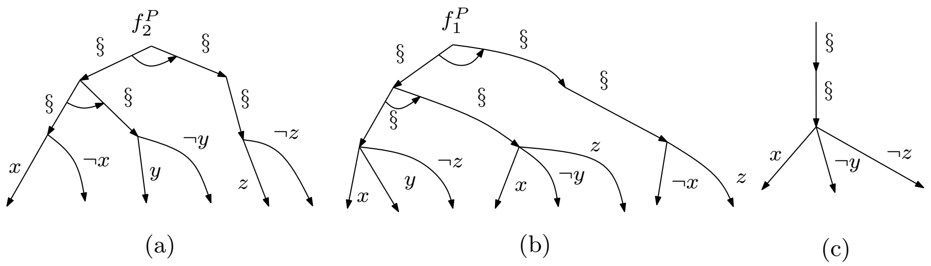


Fig. 8. (a) f_2^P. (b) f_1^P. (c) A minimal satisfier of f_2^P

In Figure 8.(c) one can find a minimal satisfier of the structural recursion, whose operational graph is given in Figure 8.(a).

Now, consider formula $P = (X \vee Y \vee \neg Z) \wedge (X \vee \neg Y \vee Z) \wedge (\neg X \vee Z)$ and the two operational graphs in Figure 8.(a)-(b). Clearly, each minimal satisfier of f_2^P gives an interpretation of the variables of P. On the other hand, minimal satisfiers of f_1^P also encode possibilities to give *true* or *false* values to the variables, here, however, X may get both a *true* and a *false* value, or it may not get any of them. All in all, it can be proven that P is a tautology iff f_1^P contains f_2^P.

Theorem 3. *The problem of containment of pure structural recursions is coNP-complete.*

7 Satisfiability and Containment Problem of Structural Recursions

The question of satisfiability is in PSPACE. Here, a very similar lemma to that of Lemma 3 can be proven.

Lemma 4. *A given structural recursion f is satisfiable iff there exists a pregraph G of U_f s.t. after polyedge reduction a constructor edge is reachable from the root in G.*

From this lemma it easily follows that the satisfiability problem of structural recursions is in PSPACE in general.

The satisfiability question is PSPACE-complete. As a quite straightforward idea in the proof we use the satisfiability problem of non-deterministic finite state automata (NDFSA), which is a well-known PSPACE-complete problem [15]. In the reduction there are three problems to be coped with. (i) Owing to the special shape of operational graphs most of the times the graph representation of an automaton cannot be taken as an operational graph immediately. In particular, an automaton may have several neighbour edges with the same label, while an operational graph has at most three, which in fact constitute a polyedge. (ii) We should represent somehow the non-deterministic nature of an automaton. (iii) The accepting states of the automaton should be represented as well.

For solving problem (i) we use a well-known construction substituting an automaton with an encoding at which every node has at most two neighbour edges with a given label. An example for this rewriting can be found in Figure 9.(a)-(b). Note that in the general case every path $a_1 \dots a_n$ in NDFSA A is substituted with a path $a_1.\S^s \dots \S^s.a_n$ in the rewriting of A ($s \geq 0$). Here, we assume that $\S$ is not contained by the alphabet of A. From now on without loss of generality we only consider NDFSA-s having at most two neighbour edges with

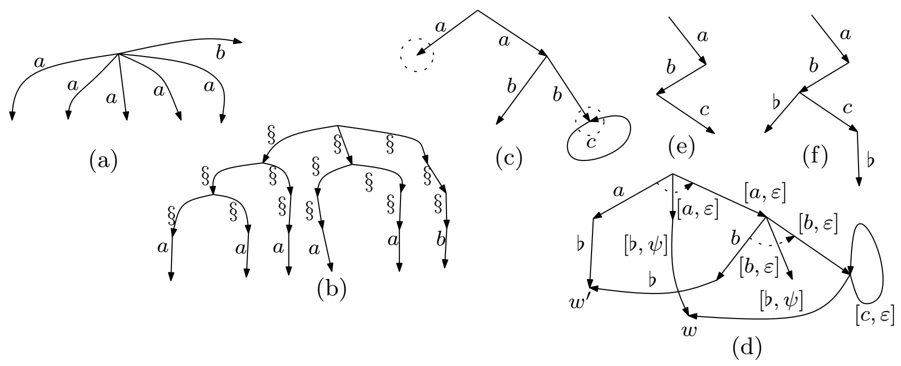


Fig. 9. (a) An NDFSA A_1 with 5 outgoing a edges. (b) The "binary representation" of A_1. (c) An NDFSA A_2. The dotted circles represent accepting states. (d) The corresponding operational graph U_{A_2} of A_2. (e) A word w accepted by A_2. (f) The data tree representation I_w of w.

the same label for every node and label. In the operational graph representation of automaton A we use the graph representation of A, and in order to solve problems (ii) and (iii) we add some new edges.

(ii) To represent the non-deterministic nature for each label a and neighbour edges e_1, e_2 having a labels, we add an $a.\flat$ path to the starting node of e_1 and e_2. This path will be the "premise branch" ending in constructor edge $\flat$ by means of which we shall be able to switch between the paths through e_1 and e_2. e_1, e_2 will be taken as the then- and else-edge respectively. Again, we assume that $\flat$ is distinct from the letters of the alphabet of A.

(iii) Finally, in order to represent accepting states, for each node corresponding to an accepting state we add a $[\flat, \psi]$-labelled constructor edge. These will be the only constructor edges in the main part of the operational graph. For an NDFSA A denote U_A its operational graph representation.

An example for this rewriting can be found in Figure 9.(c)-(d). Now, the following lemma can be proven.

Lemma 5. *Let A be an NDFSA, w an arbitrary word accepted by A and I a data graph satisfying U_A. (i) There exists a tree representation I_w of w s.t. I_w satisfies U_A. (ii) There exists a pregraph I' of I and a word w' accepted by A s.t. I' is the tree representation ($I_{w'}$) of w'.*

An example for this lemma can be found in Figure 9.(e)-(f). The following theorem is a straightforward consequence of Lemma 5 and Theorem 1.

Theorem 4. *Both the satisfiability and the containment problem of structural recursions are PSPACE-complete in general.*

Conclusions and Future Work

In the preceding sections we have discussed static analytical questions relating to structural recursions with not-isempty conditions. We have showed that the questions of satisfiability and containment are in complexity classes PTIME, coNP and PSPACE depending on whether else-branches are allowed in the conditions or not. In the next step of our research we shall examine whether our results can be applied in the research of XPath and XSLT. Since most of the static analytical questions have been already solved in this field, we are going to concentrate on XPath 2.0. We also plan to analyze these problems in the presence of single-type extended DTD-s [11], which are formal models of XML Schema Definitions.

References

1. Abiteboul, S., Buneman, P., Suciu, D.: Data on the Web: From Relations to Semistructured Data and XML. Morgan Kaufmann Publishers, San Francisco (2000)
2. XPath satisfiability in the presence of DTDs. In: Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pp. 25-36 (2005)

3. Bex, G.J., Maneth, S., Neven, F.: A formal model for an expressive fragment of XSLT. Information Systems 27(1), 21–39 (2002)
4. Breazu-Tannen, V., Buneman, P., Naqui, S.: Structural Recursion as a Query Language. In: Proceedings of the 3rd International Workshop on Database Programming Languages, pp. 9–19 (1991)
5. Buneman, P., Fernandez, M., Suciu, D.: UnQL: a query language and algebra for semistructured data based on structured recursion. The VLDB Journal 9, 76–110 (2000)
6. Buneman, P., Davidson, S., Fernandez, M., Suciu, D.: Adding Structure to Unstructured Data. University of Pennsylvania, Computer and Information Science Department (1996)
7. Garey, M.R., Johnson, D.S.: Computers and Intractability. In: A Guide to the Theory of NP-completeness. W. H. Freeman and Company, New York (1979)
8. Kósa, B., Benczúr, A.: Static Analysis of Structural Recursion in Semistructured Databases and Its Consequences. In: 8th East European Conference Proceedings, Advances in Databases and Information Systems, pp. 189–203 (2004)
9. Kósa, B., Benczúr, A., Kiss, A.: An Efficient Implementation of an Expressive Fragment of XPath for Typed and Untyped Data Using Extended Structural Recursions. In: Proceedings of the ADBIS 2009 (2009)
10. Kósa, B., Benczúr, A., Kiss, A.: Extended Structural Recursion and XSLT. Acta Univ. Sapientiae, Inform. 1(2), 165–213 (2009)
11. Martens, W., Neven, F., Schwentick, T., Bex, G.-J.: Expressiveness and complexity of XML Schema. ACM Transactions on Database Systems (2006)
12. Neven, F., Schwentick, T.: XPath Containment in the Presence of Disjunction, DTDs, and Variables. In: Proceedings of the 9th International Conference on Database Theory, pp. 315–329 (2003)
13. Milo, T., Suciu, D., Vianu, V.: Type checking for XML transformers. In: Proceedings of the Nineteenth ACM Symposium on Principles of Database Systems, pp. 11–22. ACM Press, New York (2000)
14. Suciu, D.: Distributed Query Evaluation on Semistructured Data. ACM Transactions on Database Systems 27(1), 1–65 (2002)
15. Yu, S.: Regular Languages. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, ch. 3, vol. 1, pp. 41–110. Springer, Heidelberg (1997)

Query Evaluation Techniques for Cluster Database Systems[⋆]

Andrey V. Lepikhov and Leonid B. Sokolinsky

South Ural State University, Chelyabinsk, Russia

Abstract. The paper is dedicated to a problem of effective query processing in cluster database systems. An original approach to data allocation and replication at nodes of a cluster system is presented. On the basis of this approach the load balancing method is developed. Also, we propose a new method for parallel query processing on the cluster systems. All described methods have been implemented in "Omega" parallel database management system prototype. Our experiments show that "Omega" system demonstrates nearly linear scalability even in presence of data skew.

1 Introduction

Nowadays, parallel database system applications became more and more widely distributed. Current parallel DBMS solutions are intended to process OLAP queries in petabyte data arrays. For example, DBMS Greenplum based on the MapReduce technology processes six and a half Petabyte of storage for deep analysis on a 96-nodes cluster system in the eBay [1]. DBMS Hadoop handles two and half Petabyte of storage on a cluster system consisting of 610 nodes for popular web-tool facebook. There are several commercial parallel DBMS in the area of parallel OLTP query processing among which the most known are Teradata, Oracle Exadata and DB2 Parallel Edition.

Modern researches in this area follow in a direction of self-tuning DBMS [2], data partitioning and load balancing [3], parallel query optimization [4] and an effective using of modern multicore processors [5].

One of the major objectives in parallel DBMS is load balancing. In the paper [6], it has been shown that a skews appearing in a parallel database management systems with a shared-nothing architecture at processing of a query, can lead to almost complete degradation of a system performance.

In the paper [7], an approach to solve the load balancing problem for parallel database management systems with shared-nothing architecture is proposed. This approach is based on data replication. The given solution allows reducing the network data transfer overhead during load balancing. However this approach is applicable in rather narrow context of spatial databases in a specific segment

[⋆] This work was supported by the Russian foundation for basic research (project 09-07-00241-a) and Grant of the President of the Russian Federation for young scientists supporting (project MK-3535.2009.9).

of a range queries. In paper [3], the load balancing problem is solved by partial repartitioning of the database before running a query execution. This approach reduces total amount of data transferred between computing nodes during a query processing, however it demands a very high interprocessor communication bandwidth.

In this paper, we present a new parallel query processing method based on the approach to database partitioning named "partial mirroring". This method solves problem of effective query processing and load balancing in cluster system.

The article is organized as follows. In section 2, the method of parallel query processing in DBMS for cluster system is described. In section 3, a data partitioning strategy for cluster system and algorithm of load balancing are proposed. In section 4, the results of computing experiments are shown and discussed. In conclusion, we summarize the obtained results and discuss the future work directions.

2 Organization of Parallel Query Processing

In relational database management systems with shared-nothing architecture, parallel query processing is based on data fragmentation. In this case, each relation is divided into a number of disjoint horizontal fragments, which are partitioned into different disk units of multiprocessor system. Fragmentation strategy is defined by a fragmentation function ϕ. For each tuple of the relation, this function calculates the cluster node number where this tuple has to be placed. The query is executed on all cluster nodes by *parallel agents* [8]. Each agent processes it's own fragments of the relations and generates partial query result. These partial results are merged in the resulting relation. In general case, it is necessary to perform passing tuples between the cluster nodes in order to obtain the correct result. For managing these communications, we insert the special exchange operator [9] in appropriate points of query execution plan tree.

The exchange operator is defined by a *distribution function* ψ. For each input tuple, this function calculates the number of the cluster node where this tuple has to be processed.

The exchange operator passes tuples between parallel agents by using the communication channels. Each channel is defined by the pair (node number, port number). As the node number we use the parallel agent number and as the port number we use the sequence number of the exchange operator in the query tree.

Let's describe the parallel query processing scheme in a parallel DBMS for cluster systems. We assume that the computing system is a cluster consisting from N of nodes as shown in Figure 1. We suppose the database to be partitioned into all nodes of the cluster system. According to the given scheme, SQL-query processing consists of three stages.

At the first stage, the SQL query is translated into a sequential physical plan. At the second stage, the sequential physical plan is transformed to the parallel plan being a set of parallel agents. It is obtained by inserting the exchange

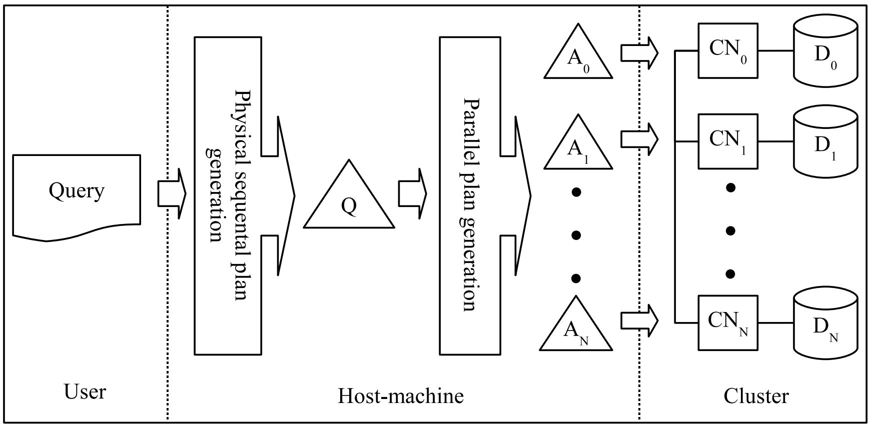


Fig. 1. The scheme of query processing in parallel DBMS for cluster systems. Q – physical query plan, A_i – parallel agent, CN_i – computing node.

operator in appropriate points of the plan tree. All parallel agents have the same structure.

At the third stage, parallel agents are spread among cluster nodes where each of them is interpreted by the query executor. Resulting relations produced by the agents are merged through the root exchange operators on the some dedicated node.

3 Data Placement and Load Balancing

3.1 Data Fragmentation and Segmentation

In the cluster system, we use the following database partitioning [10]. Each relation is divided into disjoint horizontal fragments, which are distributed among cluster nodes. We suppose that tuples in the fragment are ordered in a certain way. This order defines the sequence, in which the tuples are being read by the scan operator. We called this order as *natural order*. In practice, the natural order can be defined by a physical sequence of tuples or by an index.

At the logical level, each fragment is divided into sequence of *segments* with an equal length. The length of the segment is measured in tuples. This is a system parameter. Dividing tuples into segments is performed by the natural order beginning from the first tuple. The last segment of a fragment can be incomplete.

Let's denote the quantity of segments in fragment F as $S(F)$.

$$S(F) = \left\lceil \frac{T(F)}{L} \right\rceil .$$

Here $T(F)$ – number of tuples in fragment F, L – segment length.

3.2 Data Replication

Let fragment F_0 be allocated on disk $d_0 \in \mathfrak{D}$ in the cluster system. We suppose each disk $d_i \in \mathfrak{D}(i > 0)$ contains a *partial replica* F_i, which contains some (may be empty) subset of tuples of fragment F_0.

Segment is the smallest replication unit. The size of replica F_i is determined by the *replication factor*

$$\rho_i \in \mathbb{R}, 0 \leq \rho_i \leq 1,$$

which is a property of replica F_i. It is computed by the formula:

$$T(F_i) = T(F_0) - \lceil (1 - \rho_i) \cdot S(F_0) \rceil \cdot L.$$

The natural order of tuples in replica F_i is determined by the natural order of tuples in fragment F_0. The following formula determines the first tuple number N of replica F_i:

$$N(F_i) = T(F_0) - T(F_i) + 1.$$

If replica F_i is empty, then $N(F_i) = T(F_0) + 1$. It corresponds to "end-of-file" position.

3.3 Load Balancing Method

Parallel Agent Work Scheme. Let SQL query use n relations. Let $\mathfrak{Q}$ be the parallel plan of the SQL query. Each agent $Q \in \mathfrak{Q}$ has n input streams $s_1, \ldots,$ s_n. Each stream $s_i (i = 1, \ldots, n)$ is determined by the four parameters:

1. f_i – pointer to the fragment;
2. q_i – quantity of segments in part to be processed;
3. b_i – number of the first segment in the part;
4. a_i – load balancing indicator: 1 – load balancing is admitted, 0 – load balancing is not admitted.

An example of the parallel agent with two input streams is presented in Figure 2.

Parallel agent Q can exist in one of the two states: *active* and *passive*. In *active* state, agent Q sequentially scans tuples from the all input streams. Parameters q_i and b_i dynamically changed for all $i = 1, \ldots, n$ during scan process. In *passive* state, agent Q does not perform any actions. At the initial stage of query processing, the agent executes an initialization procedure, which defines all the parameters of the input streams and sets the agent state equal to active.

The agent begins to process fragments associated with its input streams. The agent processes only that part of the fragment, which belongs to the section defined by the parameters of the stream. When all assigned segments in all input streams have been processed, the agent turns into passive state.

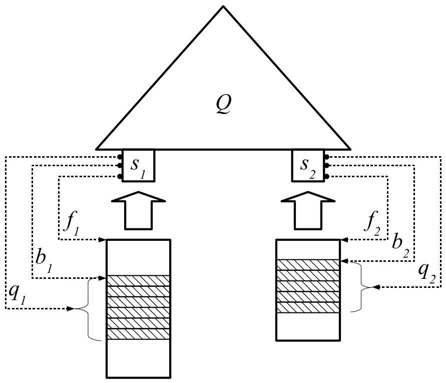


Fig. 2. Parallel agent with two input streams

Load Balancing Algorithm. During query parallel plan processing, some agents became finished while other agents of the plan continue processing segment intervals assigned to them. So we have load *skew*. To prevent the load skew, we proposes the following load balancing algorithm based on partial replication [10].

Let parallel agent $\bar{Q} \in \mathfrak{Q}$ have finished processing the segment intervals in all input streams and be turned into passive state. In the same time, let agent $\tilde{Q} \in \mathfrak{Q}$ still continue processing its segment intervals. So we have to do load balancing.

```
/* load balancing procedure between agents $\bar{Q}$ (forward)
and $\tilde{Q}$ (backward). */
$\bar{u} = Node(\bar{Q})$; // pointer to the agent node $\bar{Q}$
pause $\tilde{Q}$; // turn agent $\tilde{Q}$ into passive state
**for** (i=1; i≤n; i++) {

  **if**($\tilde{Q}.s[i].a == 1$) {
    $\tilde{f}_i = \tilde{Q}.s[i].f$; // fragment assigned to agent $\tilde{Q}$
    $\bar{r}_i = Re(\tilde{f}_i, \bar{u})$; // replica $\tilde{f}_i$ into the node $\bar{u}$
    $\delta_i = Delta(\tilde{Q}.s[i])$; // quantity of segments to trans-
fer
    $\tilde{Q}.s[i].q- = \delta_i$;
    $\bar{Q}.s[i].f = \bar{r}_i$;
    $\bar{Q}.s[i].b = \tilde{Q}.s[i].b + \tilde{Q}.s[i].q$;
    $\bar{Q}.s[i].q = \delta_i$;
  } **else**
      print("Load balancing is not permitted.");
};
activate $\tilde{Q}$ // turn agent $\tilde{Q}$ into active state
activate $\bar{Q}$ // turn agent $\bar{Q}$ into active state
```

Fig. 3. Load balancing algorithm for two parallel agents

We name the idle agent $\bar{Q}$ as *forward* and the overloaded agent $\tilde{Q}$ – as an *backward*. To prevent load skew we have transfer a part of unprocessed segments transferred from agent $\tilde{Q}$ to agent $\bar{Q}$. The scheme of load balancing algorithm is shown in Figure 3. In the algorithm, we use the *balancing function* Delta. For each stream, the balancing function calculates the quantity of segments, which have been transferred from the backward agent $\tilde{Q}$ to the forward agent $\bar{Q}$.

For effective using the described algorithm, we have to manage the following problems.

1. For each forward agent, we have to choose a backward agent, which will be the subject of balancing. A way of choosing backward agent we will name as *backward choice strategy.*
2. We have to solve, what quantity of unprocessed segments will be transferred from the backward to the forward. We will name the function calculating this quantity as *balancing function.*

Backward Choice Strategy. Let cluster system T execute parallel query plan $\mathfrak{Q}$. Let Ψ be the number of nodes of cluster T. Let forward agent $\bar{Q} \in \mathfrak{Q}$ located on the node $\bar{\psi} \in \Psi$ have finished its work. We have to choose a backward agent $\tilde{Q} \in \mathfrak{Q}(\tilde{Q} \neq \bar{Q})$ from the set of agents in parallel plan $\mathfrak{Q}$. Denote by $\tilde{\rho}$ the replication factor which defines a size of the replica $\tilde{f}_i$ for fragment $\bar{f}_i$.

To choose a backward agent, we will use ratings. In the parallel plan, we assign to each agent a rating, which is a real number. The agent with maximum positive rating has to be selected as the backward. If all the agents have negative ratings, the forward agent $\bar{Q}$ has to be over. If several agents have the maximum positive rating at the same time, we choose from them the agent, which was idle for the longest time.

The optimistic strategy uses the following rating function $\gamma : \mathfrak{Q} \to \mathbb{R}$:

$$\gamma(\tilde{Q}) = \tilde{a}_i \cdot sgn(\max_{1 \le i \le n} \tilde{q}_i - B) \cdot \tilde{\rho} \cdot \vartheta \cdot \lambda.$$

Here λ – some positive weight coefficient; B – the nonnegative integer defining the minimal quantity of segments to be transferred during load balancing; $\tilde{\rho}$ – the replication factor; ϑ – the statistical coefficient accepting one of following values:

- (-1) – the quantity of unprocessed segments for the backward is less than B;
- 0 – the backward did not participate in load balancing;
- *a positive integer* – the quantity of completed load balancing procedures, in which the backward has took a part.

Balancing Function. For each stream $\tilde{s}_i$ of backward agent $\tilde{Q}$ the balancing function Δ defines the quantity of segments, which has to be transferred to the forward agent $\bar{Q}$. In the simplistic case we can define:

$$\Delta(\tilde{s}_i) = \left\lceil \frac{\min\left(\tilde{q}_i, S(\tilde{f}_i) \cdot \tilde{\rho}\right)}{N} \right\rceil,$$

where N is the quantity of the parallel agents participating in query processing.

Function $S(\tilde{f}_i)$, introduced into the section 3.1, calculates the quantity of segments in fragment $\tilde{f}_i$. So, function Δ splits unprocessed segments of fragment $\tilde{f}_i$ into N buckets and transfers one of them from $\tilde{Q}$ to $\bar{Q}$. Such balancing function provides the uniform distribution of the load among the forward agents, which are idle.

4 Experiments

To verify the described load balancing method we performed three series of computing experiments using the "Omega" parallel database management prototype [11]. These experiments had the following aims:

- to obtain an estimations of optimum values for the parameters of the load balancing algorithm;
- to investigate the influence of the load balancing algorithm on the DBMS scalability;
- to perform the efficiency analysis of the load balancing algorithm for different kinds of the skew.

For investigation of the load balancing algorithm we used the in-memory hash-join algorithm. This algorithm is used when one of the input relations can be allocated in main memory.

4.1 Parameters of Computing Experiments

Both relations R and S have five attributes having common domain: nonnegative integers in a range from 0 to 10 million. In our experiments we processed $\theta-$join for relations R and S by the in-memory hash join method.

R was the build relation, and S was the probe relation. We selected the size of build relation R so that any fragment of R fit in main memory of the cluster node. Building hash table did not demand load balancing due to the small size of the build relation R. So we used load balancing only at the second stage of MHJ execution.

Relations R and S were created by the automated generating procedure. We used a probability model to fragment the relations on the cluster nodes. According to this model, the *skew factor* θ, $(0 \leq \theta \leq 1)$ defines the set of weight coefficients $p_i, (i = 1, \ldots, N)$

$$p_i = \frac{1}{i^\theta \cdot H_N^{(\theta)}}, \qquad \sum_{i=1}^{N} p_i = 1,$$

where N - the number of fragments, $H_N^s = 1^{-s} + 2^{-s} + \ldots + N^{-s}$ – N-th harmonic number with the degree s. Each weight coefficient p_i defines the size of $i-$th fragment in tuples.

We used one more skew factor μ to fill join attribute values in different fragments. Skew factor μ determines the percentage of "own" and "alien" tuples in the fragment. The "own" tuple should be processed in the same cluster node where it's being stored. The "alien" tuples should be transferred to another cluster node for its processing.

4.2 Investigation of Load Balancing Parameters

In the first series of experiments we investigated the following parameters: balancing interval, segment size and replication factor. The result of investigation of balancing interval is shown in Figure 4. We can see that the optimal value is 0.1 seconds. Increasing the balancing interval from 0.1 seconds to 6 seconds considerably worsens efficiency of load balancing procedure.

The result of segment size investigation is shown in Figure 5. We can see that the optimal value is 20 000 tuples. This value is approximately equal to 1% of the fragment size.

The impact of replication factor on query processing time is shown in Figure 6. We made tests for four different values of the skew factor θ. In all cases, skew factor μ was equal to 50%. We can see that the optimal value of replication factor is 0.8. In this case, the load balancing almost completely eliminates the negative impact of the skew in sizes of fragments. The full replication ($\rho = 1.0$)

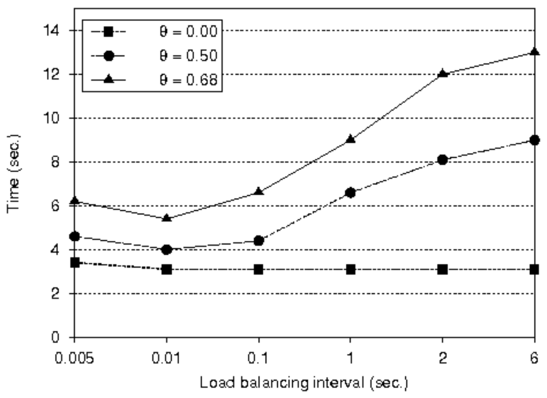


Fig. 4. Impact of balancing interval ($n = 64, \mu = 50\%$)

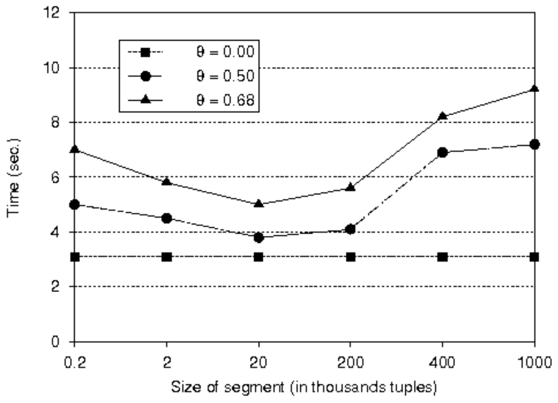


Fig. 5. Impact of segment size ($n = 64, \mu = 50\%$)

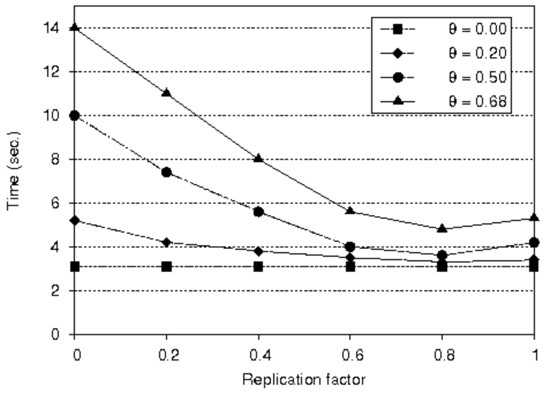


Fig. 6. Impact of replication factor ($n = 64, \mu = 50\%$)

demonstrates a worse result due to increasing the overhead of load balancing. Also, we can see that the effectiveness of load balancing grows significantly for big values of skew factor θ. If skew factor θ is equal to 0.2 (rule "25-15"), then load balancing allows us to reduce the query execution time by 30%. If $\theta = 0.68$ (rule "80-20"), then the load balancing reduces query execution time by 60%. The experiments show that the load balancing method, described in the paper, can be successfully used to eliminate a load disbalance in cluster database systems.

4.3 Scalability of Load Balancing Algorithm

In the last series of experiments, we investigate the scalability of the load balancing algorithm presented in the paper. The results of these experiments are shown on figures 7, 8 and 9.

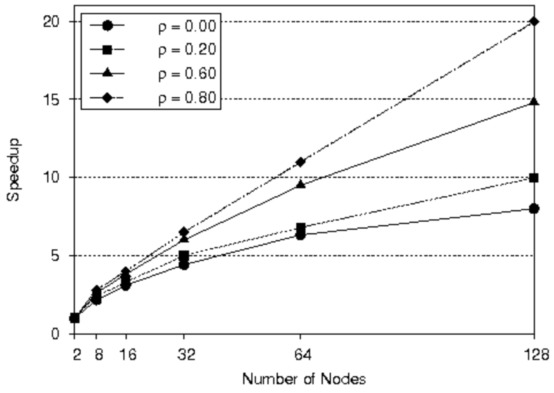


Fig. 7. Speedup versus replication factor ($\theta = 0.5, \mu = 50\%$)

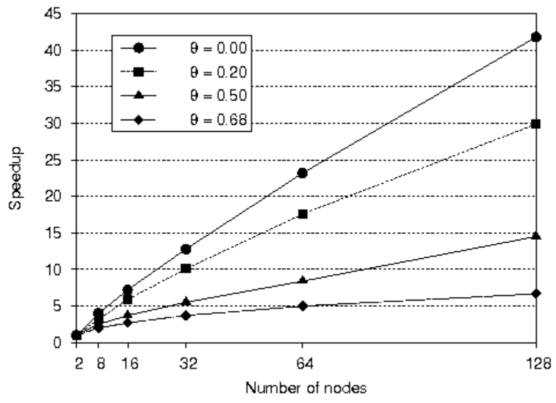


Fig. 8. Speedup versus skew factor θ ($\rho = 0.50,\ \mu = 50\%$)

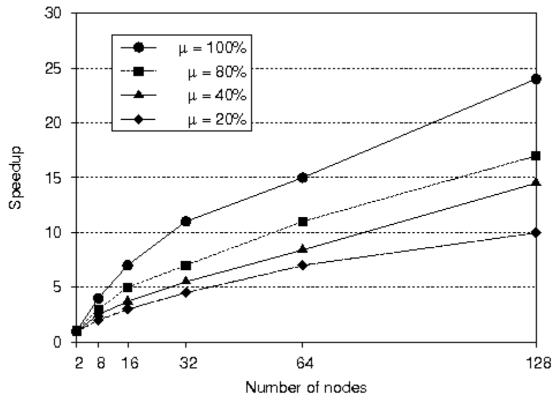


Fig. 9. Speedup versus skew factor μ ($\theta = 0.5,\ \rho = 0.50$)

In Figure 7, we presented the impact of replication factor on speedup. In this experiment, we used the following skew values: $\mu = 50\%, \theta = 0.5$. The result of the experiment demonstrates the significant growth of speedup when load balancing is being performed. Increasing the replication factor value leads to "lifting" of speedup curve. In case replication factor ρ is equal to 0.8, we have speedup near to linear.

In Figure 8, we presented the speedup curves for various values of skews factor θ. In this experiment we used replication factor $\rho = 0.5$. We can see that load balancing provides a significant speedup even at presence of severe data skew ($\theta = 0.68$ corresponds to rule "80-20").

In Figure 9, we presented the speedup curves for various values of skews factor μ. We can see that, even in the worst case $\mu = 80\%$ (80% of tuples are "alien"), load balancing provides a visible speedup.

5 Conclusions

The problem of parallel query execution in cluster systems is considered. A load balancing method and a data placement strategy are presented. Proposed approach is based on logical splitting the relation fragment by segments of equal size. The proposed load balancing algorithm is based on the method of partial replication. A strategy for choosing the backward agent is described. It uses a rating function. Proposed methods and algorithms are implemented in "Omega" DBMS prototype. The experiments at cluster system are performed. The results confirm the efficiency of proposed approach.

There are at least two directions of future work suggested by this research. First, it is useful to incorporate the proposed technique of parallel query execution into open source PostgreSQL DBMS. Second, it is interesting to extend this approach on GRID DBMS for clusters with multicor processors.

References

1. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. Communications of ACM 51(1), 107–113 (2008)
2. Chaudhuri, S., Narasayya, V.: Self-tuning database systems: a decade of progress. In: Proceedings of the 33rd International Conference on Very Large Data Bases, Vienna, Austria, September 23-27, pp. 3–14 (2007)
3. Xu, Y., Kostamaa, P., Zhou, X., Chen, L.: Handling data skew in parallel joins in shared-nothing systems. In: Proceedings of ACM SIGMOD International Conference on Management of Data Vancouver, Canada, June 9-12, pp. 1043–1052. ACM, New York (2008)
4. Han, W., Ng, J., Markl, V., Kache, H., Kandil, M.: Progressive optimization in a shared-nothing parallel database. In: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, Beijing, China, June 11-14, pp. 809–820 (2007)
5. Zhou, J., Cieslewicz, J., Ross, K.A., Shah, M.: Improving database performance on simultaneous multithreading processors. In: Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway, August 30-September 2, pp. 49–60 (2005)
6. Lakshmi, M.S., Yu, P.S.: Effect of Skew on Join Performance in Parallel Architectures. In: Proceedings of the First International Symposium on Databases in Parallel and Distributed Systems, Austin, Texas, United States, pp. 107–120. IEEE Computer Society Press, Los Alamitos (1988)
7. Ferhatosmanoglu, H., Tosun, A.S., Canahuate, G., Ramachandran, A.: Efficient parallel processing of range queries through replicated declustering. Distrib. Parallel Databases 20(2), 117–147 (2006)
8. Kostenetskii, P.S., Lepikhov, A.V., Sokolinskii, L.B.: Technologies of parallel database systems for hierarchical multiprocessor environments. Automation and Remote Control 5, 112–125 (2007)

9. Sokolinsky, L.B.: Organization of Parallel Query Processing in Multiprocessor Database Machines with Hierarchical Architecture. Programming and Computer Software 27(6), 297–308 (2001)
10. Lepikhov, A.V., Sokolinsky, L.B.: Data Placement Strategy in Hierarchical Symmetrical Multiprocessor Systems. In: Proceedings of Spring Young Researchers Colloquium in Databases and Information Systems (SYRCoDIS 2006), June 1-2, pp. 31–36. Moscow State University, Moscow (2006)
11. Parallel DBMS "Omega" official page, `http://omega.susu.ru`

Consistent Caching of Data Objects in Database Driven Websites

Paweł Leszczyński and Krzysztof Stencel

Faculty of Mathematics and Computer Science
Nicolaus Copernicus University
Chopina 12/18, 87-100 Toruń
{pawel.leszczynski,stencel}@mat.umk.pl
http://www.mat.umk.pl

Abstract. Since databases became bottlenecks of modern web applications, several techniques of caching data have been proposed. This paper expands the existing caching model for automatic consistency maintenance of the cached data and data stored in a database. We propose a dependency graph which provides a mapper between update statements in a relational database and cached objects. When update on a database is performed the graph allows detecting cached objects which have to be invalidated in order to preserve the consistency of the cache and the data source. We describe a novel method of caching data and keeping it in a consistent state. We believe that this model allows keeping the number of invalidations as low as possible. We illustrate the method using a simple web community forum application and provide some benchmarks which prove that our method is efficient when compared with other approaches.

Keywords: database caching, cache consistency, scalability, web applications.

1 Introduction

WEB 2.0 applications are data-intensive. As the number of users grows, the backend database rapidly becomes a bottleneck. Thus, various data caching techniques has been developed. Most e-commerce applications have high *browse-to-buy* ratios [9], which means that the read operations are dominant. Furthermore, such applications have a relatively high tolerance for some data to be out of date.

In this paper we show a novel method of running the cache of a data-intensive WEB 2.0 application. The goal of our research is to minimize the number of objects residing in a cache which must be invalidated after an update to the stored data. The main contribution of this paper is a coarse method to map data updates onto as small as possible sets of cached objects to be invalidated. The method uses the dependency graph composed of queries executed in order to populate the cache, objects of this cache and updates of the cached data. This graph is analysed at compile time in order to identify dependencies between cached objects and the updates. Then, at the run-time cache objects are invalidated according to the inferred dependencies. This method allows keeping the invalidations as rare as possible.

Sometimes not all select or update statements are identified at the applications development time. Some new statements can arise during the operation of the application, because e.g. they are generated at run-time by a framework. In such cases the dependency graph is augmented with new vertices and edges.

We use a web community forum application as a running example. We employ it to illustrate the whole method. It also serves as a benchmark. The experimental results prove the efficiency of our method.

The paper is organized as follows. Section 2 we present the motivating example of a forum application. In Section 3 we describe existing caching techniques and outline our consistency preserving object caching model. Section 4 we present the model in detail. Section 5 describes carried out experiments and their results. Section 6 concludes.

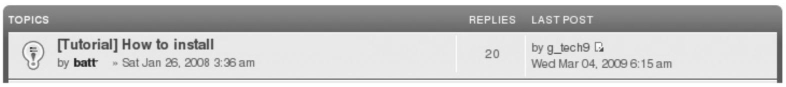
2 Motivating Example—A Community Forum Application

Let us consider a community forum application as a example. Suppose we have four tables: *user*, *forum*, *topic*, *post*. Each forum consists of topics, each of which contains a list of posts. Let us assume the following database schema:

```
user:  user_id, user_nick
forum: forum_id, forum_name, forum_desc
topic: topic_id, forum_id
post:  post_id, topic_id, post_title, post_text, user_id,
       created_at
```

When a new topic is added, its title and other data is stored in the first post. From the database's point of view the website consists of three views which are database extensive: listing forums, listing topics and listing posts. Let us now focus on listing topics. Figure 1 contains a view of a real forum application to show which data is needed when displaying list of forums.

Performing a query to display it each time the website is loaded could be extensive and harm the database. Thus such an architecture is not used in practice. Instead modern systems modify the database schema by adding redundant



TOPICS	REPLIES	LAST POST
[Tutorial] How to install by batt » Sat Jan 26, 2008 3:36 am	20	by g_tech9 Wed Mar 04, 2009 6:15 am

Fig. 1. The figure shows a single line from a list of visible topics. Each line contains: a topic name which is the first post name, an owner of the topic and the date it was created, a post count, and an information about the last post: its author and the date it was added.

data. In particular, one can add *first_post_id*,*last_post_id* and *post_count* fields to the tables *forum* and *topic* and also *topic_count* to the table *forum*.

Such a solution resolves efficiency problem stated before but also introduces new ones. It adds derived attributes to the database schema, which is discouraged in OLTP applications. Each time a new post is added, not only the *post* table needs to be modified but also *topic* and *forum*, to maintain the post count and information about latest added post. This also does not solve the main problem since the database is still the bottleneck of the system. Adding redundant data means also adding several logical constrains that have to be maintained and are error prone. It would also introduce problems when trying to replicate the whole database.

The solution is to use a cache to keep all these data in memory. However, the data are updated by the users who add posts. Whenever this happens, many post counters have to be recalculated. The desired property of a cache is to recompute only those counters which have become invalid and possibly nothing else. In this paper we show a method how to reduce the invalidations as much as it is practically possible.

3 Caching

3.1 Existing Caching Algorithms

Before the WEB 2.0 era several techniques of caching whole websites have been proposed and surveyed in [14,18]. They construct mappers between URLs and database statements. However this approach loses efficiency in WEB 2.0 applications since the same HTML can used on many pages and it does not make sense to invalidate the whole pages.

The existing caching models can be divided into caching single queries, materialized views and tables. When caching single queries it may be hard to discover similarities and differences between the queries and their result. Let us consider two similar queries which return the same result set, but the first one produces it in ascending order, while the second one emits the descending order.

Caching based on hash based algorithms does not work well in the presented application and applies more to caching files than to the OLTP database schemas. The idea of caching tables' fragments has been first shown in [5] where authors propose a model with dividing tables into smaller fragments. It can be understood as storing data sets in caches and allowing them to be queried [21,8,9]. However this does not solve the whole problem since it lacks count operations. In the forum example and most WEB 2.0 applications the website contains several counters which cannot be evaluated each time the website is loaded. Performing count operations on the cached data sets is difficult because of the complexity of detecting that all data to be counted is loaded to the cache. There is also no need to store whole data when only counters are needed. The data can be loaded ad hoc or loaded dynamically each time it is needed. When all data is loaded to cache at once, one can easily discover which count queries can be performed but it also means caching data that may never be used. Also when an update occurs all cache needs to be reloaded. This may cause severe problems because updates occur frequently in OLTP. This also means performing updates to maintain the consistency of the cached data which is never used but stored in cache. Invalidation can occur each time the update occurs or in the specified time intervals [13]. The second case would be efficient but would also allow storing and serving data that is not up to date. Loading data statically is more like database replication than a caching technique. An interesting recent work on replicating data sources and reducing communication load between backend database and cache servers has been described in [6]. It presents an approach based on hash functions that divide query result into data chunks to preserve the consistency. However this also does not solve the problem of aggregation queries whose results are difficult to be kept consistent via hash similarity.

Authors of [1,2,3] present model that detects incosistency based on statements' templates which is similar to the presented model. However their approach cannot handle join or aggregation in select statements. Our approach is based on a graph which edges determine the impact of the update statements on the cached data. The idea of the graph representation has been first presented in [10,11,12]. The vertices of the graph represent instances of update statements and cached data objects. However nowadays most webpages are personalized and number of data objects has increased and multiplied by a number of application users. According to these observations the graph size can grow rapidly and the method becomes impractical. In our approach the dependency graph has vertices which represent classes of update and select statements and not individual objects. This idea allows reducing the number of vertices.

The problem can be understood as finding the best trade-off between efficiency and consistency for the given application. Described schemata show that data granularity of the cached data is strongly desired. In that case only atoms which are not up to date would be invalidated thus improving the efficiency. However these atoms cannot be understood as table rows, since count would be difficult to define, and they should be more like tuples containing data specified by the application logic. Schemata shown above cannot afford it because of persistent proxy between application server and database. On one hand this feature aids software programmers because they do not need to dig into caching techniques. On the other hand it is the software programmer who has to specify what data has to be cached because it is strongly related to the application's specific logic.

3.2 Caching Objects vs. Caching Queries

Most of previously described caching techniques include caching queries. This solution needs a specification of queries that need to be cached because they are frequently performed. If a query used for listing topics from the community forum is taken into consideration, one can argue if it makes sense to cache its result. On one hand the query is performed each time the user enters a specific forum but on the other hand one should be aware of user conditions. If the user searches for topics with a post containing a specific string it may be useless to cache them because of the low probability they will ever be reused.

Instead of caching queries one should take into consideration caching objects. Suppose objects of class *topic* and *forum* are created and each of them contains the following fields:

```
FORUM: forum_id, forum_name, post_count, topic_count,
       last_post_author_nick, last_post_created_at
TOPIC: topic_id, first_post_title, first_post_created_at,
       first_post_author_nick, last_post_author_nick,
       last_post_created_at
```

Having such objects stored in a cache, the query listing topics could be the following:

```
SELECT topic_id FROM topic WHERE topic_id = $topic_id
AND ## user condition LIMIT 20;
```

When having list of id's of topics we simply get those objects from the cache. Each time the object does not exist in the cache it is loaded from the database and created. This means significant reduction of the query complexity and the performance improvement. *Memcached* [15] is an example of the mechanism widely used in practice. It is high-performance, distributed memory object caching system and is used by *Wikipedia*, *Facebook* and *LiveJournal*. In December 2008 *Facebook* considered itself as the largest *memcached* user storing more than 28 terabytes of user data on over 800 servers [17].

From theoretical point of view the idea can be seen as using a dictionary for storing data objects created from the specified queries. One can argue if storing relational data inside the dictionary is sensible. Here the performance becomes an issue. Since all the cached data is stored in RAM (it makes no sense to cache data on a disk) a cache server only needs to hash the name of the object and return data which is stored under the hashed location. The caching mechanism is outside the database which means a significant performance gain since the database workload reduced.

3.3 Data Consistency Problem

Data consistency problem arises when caching techniques are applied. Let us assume *topic* objects in the forum example are cached. Suppose one software programmer has created functionality of caching objects and few months later the other is asked to add functionality of editing user's *user_nick*. Each time *user_nick* is changed all objects of topics owned by this user or topics where user has his last post need to invalidated. How can the programmer from one department know about the functionality made by the other one which is hidden in the data model? According to [19] *Wikipedia* uses a global file with a desrciption of all classes stored in a cache, the file is available on [20]. However this is not a general solution to the problem but only an improvement which helps programmers to manually make a safe guard from using objects in an inconsistent state. In this paper we propose the model of fully automatic system which maintains the consistency of cached data.

4 The Dependency Graph

4.1 Basic Assumptions

As described before data granularity is strongly needed in the caching systems. However when caching objects, it is not possible to track the dependencies between all possible update statements and objects. The graph will track only the dependencies within the database schema. It assumes update statements are performed on single rows identified by primary keys and cached objects are identified by the class and the primary key of the specified relation.

The following assumptions restrict the syntax of SQL facilitated by our method. This restriction is needed since mapping the relational data to the dictionary is performed.

1. Select statements are significantly more frequent than inserts and updates.
2. Database consists of k relations.
3. Each table's primary key consist of a single attribute.
4. One can identify a set of select statements $S = \{S_1, S_2, ..., S_r\}$ which are used for creating cached objects.
5. Set $U = \{U_1, U_2, ..., U_m\}$ is a set of queries which modify the database and each of them modifies only single row in a table reached by its primary key. Additionally select statement can have other conditions in the WHERE clause but they can only involve columns of the parameterised table.
6. Cached objects and queries from S and U are parameterised by the primary key of some relation.
7. For each class of the object we can identify subset of queries from S used to create the object.
8. System does not assume anything about data in cached objects, but only knows select statements used to create them.

Sometimes it is convenient to know cached data to optimise invalidation clues. However the model assumes data inside objects to be persistent because sometimes instead of caching data developers decide to cache whole HTML fragments corresponding to the cached objects. Other database caching systems could not allow this because of being persistent to the application server. Once again the persistence of caching models reveals its drawback. The presented model can be also seen as conjunction of caching static HTML pages, which mechanisms are clearly described in [7], and data from database.

4.2 Query Identification

Let us first identify queries used by the application when creating objects. List of queries used when creating topic object follows:

```
S1:   SELECT * FROM topic WHERE topic_id = $topicId;
S2_1: $max = SELECT max(p.created_at) FROM post p, topic t WHERE
      t.topic_id = p.topic_id AND t.topic_id = $topicId;
```

```
S2_2: SELECT u.user_nick FROM post p, user u, topic t WHERE
      t.topic_id = p.topic_id AND t.topic_id = $topicId AND
      p.user_id = u.user_id AND p.created_at = $max;
S3_1: $min = SELECT min(p.created_at) FROM post p, topic t WHERE
      t.topic_id = p.topic_id AND t.topic_id = $topicId;
S3_2: SELECT u.user_nick FROM post p, user u, topic t WHERE
      t.topic_id = p.topic_id AND t.topic_id = $topicId AND
      p.user_id = u.user_id AND p.created_at = $min;
S4:   SELECT count(p.post_id) FROM post p, topic t WHERE
      p.topic_id = t.topic_id AND t.topic_id = $topicId
```

The first statement gets a row from a *topic* table. $S2\_1, S2\_2$ and $S3\_1, S3\_2$ are very similar and are used for getting data of the first and the last post in the topic. Additionally $S4$ is performed to evaluate a number of posts in the topic.

4.3 Dependency Graph Construction

In this section a creation of the dependency graph is described. As before in the model description let us assume k relations R in the database schema and define the set of attributes for each relation. Additionally let O denotes the set of classes of objects stored in a cache

$$Attr(R_1) = \{r_{11}, ..., r_{1m_1}\}, ..., Attr(R_k) = \{r_{k1}, ..., r_{km_k}\} \tag{1}$$

$$R = Attr(R_1) \cup Attr(R_2), ... \cup Attr(R_k) \tag{2}$$

$$O = \{O_1, O_2, ..., O_n\} \tag{3}$$

Vertices The dependency graph consists of the vertices from sets O, R, S, U where S and U are sets of select and update statements defined before. The vertices of the graph form four layers as presented on Figure 2. Each ellipse denotes vertices of attributes in a relation but only attributes are vertices of the graph.

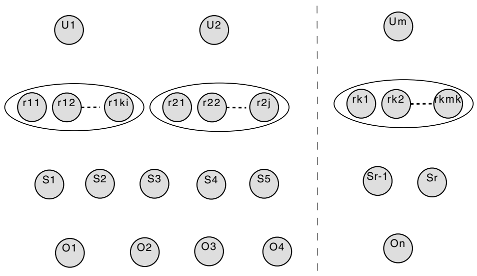


Fig. 2. Vertices of the dependency graph

Edges As vertices are defined let us now construct edges in the graph. There are two kinds of edges: weak and strong. Weak edges are used to find object to be invalidated, while strong edges are used to identify keys of these objects.

1. *Edges with vertices from U*: As assumed before, each update performs modification of a single row identified by a primary key. We create edges from update vertex U_i to all attributes that have been modified (Fig. 3). Especially if a new row is inserted or existing one is deleted then all attributes are modified and edges to all vertices are added. Additionally we will distinguish between strong and weak edges and the edge connecting attribute of the primary key are strong while the other are weak.

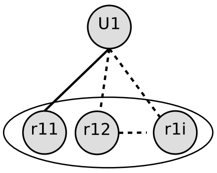


Fig. 3. Edges corresponding to update statement U1. Strong edges are solid, while weak edges are dotted.

2. *Edges between attributes* are between the primary key and all other attributes within a relation and also between the primary key attribute and its corresponding foreign keys (see Figure 4). All edges between attributes are strong.

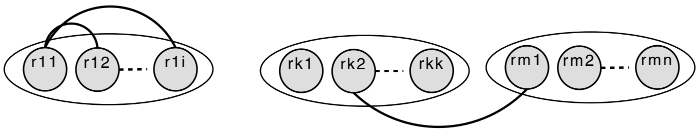


Fig. 4. Edges between attributes

3. *Edges for select queries*: The simplest case is when single row in a table is accessed via primary key. The left-hand side of Figure 5 shows this. The edges connect the query node to all selected attributes and the attributes used in the WHERE clause. The edge containing primary key of the relation is strong.

   On the right of Figure 5 one can see edges for a join query. Only one-to-one and one-to-many joins are facilitated by our method. It means that join can be performed only using a primary key. Attributes that are used within the selection are connected by an edge with the query vertex.

   In our running example four queries: $S1, S2\_1, S2\_2, S3\_1, S3\_2$ and $S4$ have been identified for creating the *topic* object. The $S1$ statement returns single row of the table identified by the primary key. $S4$ is a select statement with join on one to many relation and is allowed by the model. The $S2\_1$, $S2\_2$

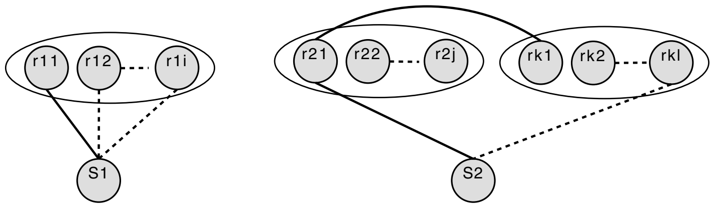


Fig. 5. Selection queries: a single point select query and a join querie

and $S3\_1, S3\_2$ look quite similar and are used for retrieving data of the first and the last post. $S2\_1$ and $S3\_1$ are select statements with the single join on one to many relation. $S3\_1$ and $S3\_2$ include two joins but the last condition in the WHERE clause does not correspond to the parameterised table. Both statements are parameterised with the *topic* primary key and the condition uses columns from the *post* table which is ruled out by the restrictions imposed in the model. In that case the caching system treats such a condition as non-existent. The system is then still correct, since it will not miss any object invalidation. However, it can sometimes unnecessarily invalidate objects normally filtered out by the uncared for condition.

4. *Edges mapping objects to queries* The lowest layer of the graph connects cached objects with select statements used for creating them and only those statements need to be added to the graph. Cached objects are parameterised by the primary key attribute and model works under the assumption that only this parameter can be used in the select statements as the condition.

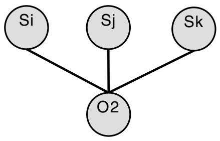


Fig. 6. Select statements used for creating objects

4.4 Forum Example

In the community forum example update statements need to be identified. We can find five statements that manipulate data:

```
U1: INSERT INTO user VALUES ...
U2: INSERT INTO forum VALUES ...
U3: INSERT INTO topic VALUES ...
U4: INSERT INTO post VALUES ...
U5: UPDATE user SET nick WHERE user_id = ...
```

In Section 4.2 we listed select statements used to retrieve a topic object. Since we have the updates and the queries we can create the dependency graph. Figure 7 shows the graph of the running example.

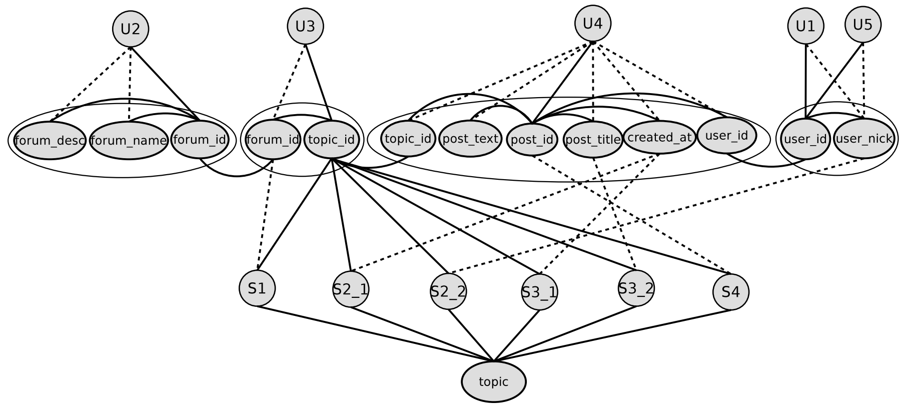


Fig. 7. A graph created for a topic objects in the community forum example

4.5 The Algorithm

The presented algorithm relies on the dependency graph. It can be statically generated at the system set up. On one hand it can be assumed that the graph is created before the web application runs. However this assumption can be false in some cases. In most modern development frameworks programmers do not write SQL statements by hand. These statements are automatically generated from the database schema. In such a case it is impossible to list upfront the queries used by the application. Therefore, the dependency graph needs to be created dynamically with respect to performed statements.

Even if some graph elements will be identified when a web application runs, many of them are known in advance. Attribute vertices and edges between them can be created from the database schema.

```
static G := initial_graph(database schema)
```

The rest of algorithm can be divided into two separate parts: the action for an update statement and the action for a select statement. When select statements are performed, the system checks if new vertices and edges need to be added to the graph. This routine only checks if an augment to the graph is needed. No object gets invalid. Therefore, this routine is not performed when the graph is static, i.e. no new select statements can appear at run-time. The following code snippet shows the routine for select statements:

```
validateSelect(stmt, class)
{
  if (object_vertex(class) is null) V(G) += {class}
  v := statement_vertex(stmt)
  if (v is null)
     V(G) += {stmt};
     edges := {weak_edges(stmt), strong_edges(stmt)};
     E(G) += edges;
}
```

Given the statement and the class of the cached objects we need first to check if the vertex corresponding to the object class exists. If not it is added to the graph. Then the vertex of the performed select statement is being found. If it does not exist it is being added. In that case additionally weak and strong edges need to be identified. Weak edges are easily identified since they bind the columns used by the selection. On the other hand strong edges between the attributes have been added on the system set up and only the edges between attributes, select statements and cached object need to be detected. This is however easy since SQL statement is parameterised only via one parameter.

Let us now focus on the action for update statements. Again, if all update statements are known at the compile time, the graph updating part (lines 6–9) of the following routine need not to be performed.

```
1.  validateUpdate(stmt)
2.  {
3.    param := statement_parameter(stmt);
4.
5.    v := statement_vertex(stmt);
6.    if (v is null)
7.       V(G) += {stmt};
8.       E(G) += edges_of_update_statement(stmt);
9.
10.   ov := objects_reached_by_path_three(v);
11.
12.   foreach (o in ov)
13.      spath := shortest_path(v,o);
14.      params := [param];
15.      prev := v;
16.      while (next := next_node_in_path(spath, prev))
17.         params := next_vertex_params(params, prev, next);
18.         prev := next;
19.      invalidate_objects_of_class(o, params);
20.  }
```

When new update statement is received we first need to check if the corresponding vertex exists. If not it is being added with all edges: one strong edge to the primary key attribute and weak edges to all attributes of the modified columns. Then, at line 10, the system identifies classes of objects that may need an invalidation. It searches for the paths of length 3 between the given update vertex and vertices of object classes using both: strong and weak edges. The length 3 is important since it means that there is a column modified by the update and used by some select statement. This can be also seen on the example graph (Fig. 7). When a new forum is added there exists a path between $U2$ and *topic* vertex but *topic* object needs not to be invalidated since no column used to create it have been changed.

Having found the classes which may contain invalid objects the algorithm finds those invalid instances. For each object vertex the system searches for the shortest path between the update vertex and the object node but only using strong edges. Weak edges are used to find vertices of object classes, while the strong edges are applied for getting keys of those objects. For each object vertex the system goes through the shortest path and gathers parameters of reached vertices. Eventually it stops in the object vertex with the list of cached objects parameters. Those objects instances are removed from the cache. This time when going through the path the algorithm uses only the strong edges because they allow to gather parameters of next vertices.

Let us consider statement *U4* (adding new post). The system starts from node *U4* and goes through the *post_id* attribute to the *topic_id* in the *post* and the *topic* table. Having the *topic_id* it knows the exact key of the object to invalidate since topic object and select statements are parameterised by the same primary key. In this case no additional queries need to be performed on a database.

However, in case of *U5* (updating a user's nick), having a *user_id* the system needs to find all posts written by the user. This means a query to the database must be performed. The system invalidates then all topics where user has written posts. This can be seen as a drawback but it is impossible to examine if a user's post is really the first or last without querying the database. One can argue if it can be improved for *min*() and *max*() functions but it surely cannot be done for *avg*() so no general solution without digging into SQL syntax exists.

The other thing is the chain of joins between tables. If the shortest path goes through several joins which require querying database the system can be inefficient. However, it applies well in most cases since in OLTP queries shall be kept as simple as possible. One should also resemble that even having complicated database schema not all of data has to be cached and objects should be kept as granular as possible to prevent extensive invalidation.

5 Experiments

The presented model has been tested on a RUBiS benchmark [22]. RUBiS is an auction site prototype modelled after *www.ebay.com* and provides a web application and a client emulator. The emulator is modelled according to a real life workload and redirects the client from one web page to another due to a predefined transition probability. In the presented benchmark we have used 100 client threads running at the same time for 18 minutes.

We have tested the application in different modes: without caching, caching with a defined *time-to-live* of the cached objects and caching with the invalidation management based on the dependency graph. The application benchmark has been run 3 times in each mode. Figures 8 and 9 display achieved results. Our consistency preserving model reduces up to 54% of performed queries. It is more efficient than techniques based on *time-to-live* and does not store stale data. In the experiment no cached objects have been invalidated when unnecessary and there have been no queries that did not fit to the SQL syntax restrictions set by our model.

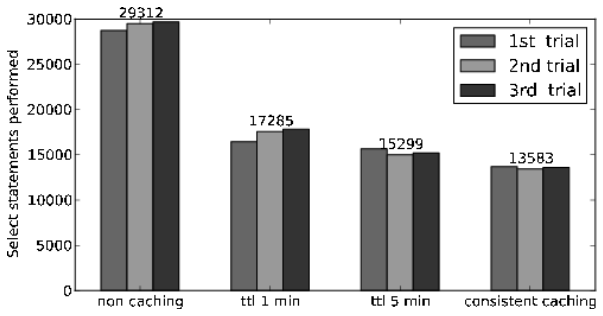


Fig. 8. The comparison of different caching techniques. The numbers indicate average count of select statements for each technique.

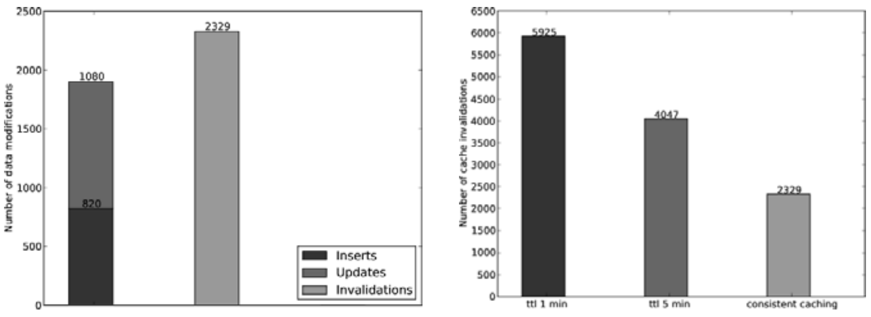


Fig. 9. (a) Number of data modifications. (b) Number of cache invalidations in different models.

We have also measured a number of data modifications and the number of cache invalidations as stated in a figure 9. In the presented benchmark 7.1% of database statements have modified data and none of them updated more than one row. Our assumption that we deal with the read dominant database communication is therefore correct. The left part of the figure shows that the number of cache invalidations does not grow rapidly when compared to database statements. The right figure shows that almost 61% cache invalidations can be saved by the presented model when compared to *time-to-live* techniques which proves the significant improvement of the presented model to those technique.

6 Conclusion

The database bottleneck is the most serious problem in modern web applications which is solved in practice by providing a scalable *key-value* cache storage engines. This however causes the consistency problem between a relational database and a cache. In this paper we presented a novel cache model based on the dependency graph which detects invalidations of the cached objects when updates occur.

We provided series of tests on the RUBiS benchmark. We observed that restrictions on SQL introduced in the model are not harmful in the context of web applications. We observed a significant reduction of performed database statements.

The presented approach is more efficient than *time-to-live* techniques and does not allow serving data which is not up to date. When compared to the template approach several improvements need noting. First, we allow join and aggregation in select statements which is very important since many aggregation functions are used in the modern web applications to provide frequent counters displayed on websites. Second, template based approaches need to know all performed statements classes in advance since the evaluation of invalidation rules is time consuming. Our dependency graph can be easily updated at any time since adding or removing vertices does not require complex operations. When compared to materialized views our mechanism does not exploit any knowledge of cached data and its structure. Also note that materialized views reside on the database side and thus they cannot solve the database communication bottleneck.

Since the invalidation policy does not rely on the cached data and its structure, it allows storing semi-structured data. The future work may involve caching whole HTML code fragments. This can be also understood as an interesting consistency mapper between database and websites components for storing the current HTML. We also consider integration of the algorithm with one of the existing Object Relational Mappers. This could be also extended to an automatic generation of cached objects and invalidation rules due to a predefined database schema.

References

1. Garrod, C., Manjhi, A., Ailamaki, A., Maggs, B., Mowry, T., Olston, C., Tomasic, A.: Scalable query result caching for web applications. In: Proceedings of the VLDB Endowment Archive, vol. 1(1), pp. 550–561 (2008)
2. Garrod, C., Manjhi, A., Ailamaki, A., Maggs, B., Mowry, T., Olston, C., Tomasic, A.: Scalable Consistency Management for Web Database Caches. Computer Science Technical Reports, School of Computer Science. Carnegie Mellon University (2006)
3. Manjhi, A., Gibbons, P.B., Ailamaki, A., Garrod, C., Maggs, B., Mowry, T.C., Olston, C., Tomasic, A., Yu, H.: Invalidation Clues for Database Scalability Services. In: Proceedings of the 23 rd International Conference on Data Engineering (2006)
4. Choi, C.Y., Luo, Q.: Template-based runtime invalidation for database-generated Web contents. In: Yu, J.X., Lin, X., Lu, H., Zhang, Y. (eds.) APWeb 2004. LNCS, vol. 3007, pp. 755–764. Springer, Heidelberg (2004)
5. Dar, S., Franklin, M.J., Jónsson, B.P., Srivastava, D., Tan, M.: Semantic Data Caching and Replacement. In: Proceedings of the 22th International Conference on Very Large Data Bases Table of Contents, pp. 330–341 (1996)
6. Tolia, N., Satyanarayanan, M.: Consistency-preserving caching of dynamic database content. In: Proceedings of the 16th International Conference on World Wide Web, pp. 311–320 (2007)

7. Katsaros, D., Manolopoulos, Y.: Cache Management for Web-Powered Databases. Encyclopedia of Information Science and Technology (I), 362–367 (2005)
8. Altnel, M., Bornhvd, C., Krishnamurthy, S., Mohan, C., Pirahesh, H., Reinwald, B.: Cache tables: Paving the way for an adaptive database cache. In: Proc. VLDB 2003, pp. 718–729 (2003)
9. Luo, Q., Krishnamurthy, S., Mohan, C., Pirahesh, H., Woo, H., Lindsay, B., Naughton, J.: Middletier database caching for e-business. In: Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, pp. 600–611 (2002)
10. Iyengar, A., Challenger, J., Dias, D., Dantzig, P.: High-Performance Web Site Design Techniques. IEEE Internet Computing (4), 17–26 (2000)
11. Challenger, J., Dantzig, P., Iyengar, A., Squillante, M.S., Zhang, L.: Efficiently Serving Dynamic Data at Highly Accessed Web Sites. IEEE/ACM Transactions on Networking 12, 233–246 (2004)
12. Challenger, J., Iyengar, A., Dantzig, P.: A Scalable System for Consistently Caching Dynamic Web Data (1999)
13. Zhao, W., Schulzrinne, H.: DotSlash: Providing Dynamic Scalability to Web Applications with On-demand Distributed Query Result Caching, Computer Science Technical Reports, Columbia University (2005)
14. Katsaros, D., Manolopoulos, Y.: Cache management for Web-powered databases. In: Web-Powered Databases, pp. 201–242 (2002)
15. Memcached, Danga Interactive, http://www.danga.com/memcached/
16. Velocity, http://code.msdn.microsoft.com/velocity
17. Scaling memcached at Facebook, http://www.facebook.com/note.php?note_id=39391378919
18. Li, W., et al.: CachePortal II: Acceleration of very large scale data center-hosted database-driven web applications. In: Proc. VLDB (2003)
19. Managing Cache Consistency to Scale Dynamic Web Systems; Chris Wasik; Master thesis at the University of Waterloo (2007), http://uwspace.uwaterloo.ca/handle/10012/3183
20. memcached.txt in Wikipedia, http://svn.wikimedia.org/viewvc/mediawiki/trunk/phase3/docs/memcached.txt?view=markup (04.04.2009)
21. Amiri, K., Tewari, R.: DBProxy: A Dynamic Data Cache for Web Applications. In: Proc. ICDE, pp. 821–831 (2003)
22. RUBiS (Rice University Bidding System), http://rubis.ow2.org/

An Ontology Driven Approach to Software Project Enactment with a Supplier

Miroslav Líška[1] and Pavol Návrat[2]

[1] DATALAN, a. s., Galvaniho 17A, 821 04 Bratislava, Slovakia
[2] Faculty of Informatics and Information Technologies,
Slovak University of Technology in Bratislava,
Ilkovičova 3, 842 16 Bratislava, Slovakia
miroslav_liska@datalan.sk, navrat@fiit.stuba.sk

Abstract. SPEM is metamodel based standard used to define software and systems development processes and their components. Unfortunately, its architecture is semiformal, thus it is not possible to make and to verify created language statements with formal techniques such as the consistency or satisfiability verification. Recently, the combination of MDA and the Semantic Web, in which data processing is concerned with regard to their semantics, become the leading subject in this direction. In this work we present a SPEM transformation to the Semantic Web technical space and consequently we propose its utilization that is an ontology based approach to software project enactment with a supplier. We discuss its usage scenarios that are a verification of a set of SPEM methods and processes with ontology, and a project plan generation and verification with a set of SPEM method plugin ontologies. Additionally we present examples that addresses to the proposed usage scenarios.

Keywords: MDA, SPEM, OWL, Semantic Web, project plan verification, software project enactment.

1 Introduction

The difficulty of software development is greatly enhanced when it is inevitable to cooperate with a supplier. The general issue is to manage a lot of differences such as different tasks, software work products, guidelines, roles etc. (IEEE 2004). The ideal state is that a company and its supplier use the same software framework and they use it equally. Otherwise risk of budget and time overrun together with quality fall are greatly increased. Unfortunately the ideal state that we have mentioned cannot exist. Either companies use different software frameworks, or they use the same software framework but necessarily differently. It is natural that companies have different knowledge obtained from their various projects, and also have different peoples with different experiences. Thus even they use the same software framework, e.g. Rational Unified Process (Kruchten 2003), the project enactment is problematic. One of the problems is that standard software development process frameworks are usually used as a navigable websites that contain only human-readable descriptions. This fact allows existence of undesirable ambiguities inevitably. Thus, these kinds of frameworks cannot be used to represent machine interpretable content (Zualkernan 2008). Moreover, these frameworks are used in the technical spaces (Kurtev 2002) that have model based architecture, such as the Model Driven Architecture (MDA) or the Eclipse Modeling Framework (EMF) (Steinberg 2009). These kinds of technical spaces also limit knowledge based processing, owing to their weakly defined semantics (Gašević 2009).

However, at present the emerging field of Semantic Web technologies promises new stimulus for Software Engineering research (Happel 2006). The Semantic Web is a vision for the future of the Web, in which information is given explicit meaning, making it easier for machines to automatically process and integrate information available on the Web (Mcguinness 2004). The today's key Semantic Web technology is the Web Ontology Language (OWL). OWL is intended to be used when the information contained in documents needs to be processed by applications, as opposed to situations where the content only needs to be presented to humans (Smith 2004).

Thus if we transform a definition of software methods to the technical space of the Semantic Web, we can use many knowledge oriented techniques to maintain them. In this work we address such an opportunity. We propose an approach to software method specification in technical space of the Semantic Web and consequently we propose its inherent utilization that is an ontology based software methods integration. Our goal is to use this ontology oriented method for project enactment with a supplier in the context of the SWEBOK (IEEE 2004). We use SPEM, the MDA standard used to define software and systems development processes and their components (OMG 2008). We transform SPEM from the MDA technical space into the Semantic Web technical space; so we can work with SPEM as with an ontology.

1.1 Related Works

Usability of ontologies in software engineering and technology (SET) can be distinguished into two main categories: SET domain ontologies and ontologies as software artifacts (Calero 2006). SET domain ontologies refer to the ontologies whose main goal is to represent (at least partially) knowledge of a certain subdomain within SET matter. In the second category, ontologies are used as software artifacts of diverse types, in some software process. Based on the SWEBOK guide, prototypes of ontologies for the representation of the complete software engineering domain have been created (Mendes 2005, Sicilia 2005). Other ontology that also conceptualizes the software engineering domain, is OntoGLOSE (Hilera 2005), created and based on the "Glossary of Software Engineering Terminology" published by the IEEE (2002). Falbo et al. (1992) and Larburu et al.(2003) have proposed ontologies to model the knowledge related to the software process, including concepts such as Life Cycle Model, Software Process, Activity, Procedure, Task, Role, or Artifact, among others. Since we want to use SPEM in the technical space of the Semantic Web and SPEM is MDA based, we can utilize research results that concern with using MDA in the technical space of the Semantic Web.

SPEM is specified in the Meta Object Facility (MOF) language that is the key language of MDA. MOF is a language for metamodel specification and it is used for specification of all model-based MDA standards (Frankel 2003). It provides metadata management framework, and a set of metadata services to enable the development and interoperability of model and metadata driven systems (OMG 2006). On the Semantic Web side, OWL is intended to provide a language that can be used to describe the classes and relations between them that are inherent in Web documents and applications. OWL is based on Resource Description Framework Schema (RDFS) (Brickley 2004). Both MOF and RDFS provide language elements, which can be used for metamodeling. Although they have similar language concepts such as mof:ModelElement with rdf:Resource, or mof:Class with rdf:Class, these languages are not equivalent. RDFS, as a schema layer language, has a non-standard and non-fixed-layer metamodeling architecture, which makes some elements in model to have dual roles in the RDFS specification (Pan 2001). MOF is also used for specification of the Unified Modeling Language (UML) that is a language for specification, realization and documentation of software systems (OMG 2009a). Even if UML and RDFS are similar in a domain of system specification, they are also substantially different. One issue that has been addressed was the problem that RDF properties are first class entities and they are not defined relative to a class. Therefore a given property cannot be defined to have a particular range when applied to objects of one class and another range when applied to objects of a different class (Cranefield 2001). Note this difference have also been propagated between OWL and UML (Hart 2004). At present the main bridge that connects the Semantic Web with MDA is stated in the Ontology Definition Meta-Model (ODM) (OMG 2009b). ODM defines the OWL Meta-Model specified in MOF (MOF - OWL mapping) and also the UML Profile for Ontology modeling (UML - OWL mapping). This architecture can be extended with additional mappings between the UML Profile for OWL and other UML Profiles for custom domains (Gašević 2005, 2006). In our previous work we have utilizied this principle and created an approach of SPEM UML model validation in SPEM Ontology (Líška 2009) and approach of project plan generation and verification with SPEM Ontology (Líška 2010a).

An approach that is close to the topic of this work which uses SPEM in the Semantic Web technical space proposes the use of defining SPEM process constraint with the Semantic rules with SWRL (Rodríguez 2009). SWRL is W3C Semantic Web Rule Language that combines OWL and RuleML (Horrocks 2004). The paper proposes OWL-DL consistency reasoning between a software process ontology and a project plan, where SWRL rules can be used to define additional logical constraints. The second approach (Zualkernan 2008) that also uses SPEM in the domain of Semantic Web presents a competency framework for software process understanding. The paper describes an ontology and a system that automatically generates assessments for the SCRUM (Schwaber 2002) engineering process.

1.2 Aims and Objectives

We aimed in our research to devise a transformation of SPEM into the Semantic Web technical space; and an ontology based approach to software project enactment with a supplier. Besides that, we present an example of the enactment with formal model, where the subjected software methods are two different requirements specification work definitions.

The rest of the paper is structured as follows. Section 2 presents our solution to the problem. First we define a conformance level with the SPEM compliance point, and also we present a SPEM transformation into the Semantic Web technical space. Then we describe our approach to ontology based software project enactment with a supplier. Subsequently, Section 3 presents an example of usability of such method with formal model. Finally, Section 4 provides conclusion and future research direction.

2 Approach

SPEM is MDA standard used to define software and systems development processes and their components. The scope of SPEM is purposely limited to the minimal elements necessary to define any software and systems development process, without adding specific features for particular development domains or disciplines (e.g., project management). SPEM metamodel is MOF-based and reuses UML 2 Infrastructure Library (OMG 2009c). Its own extended elements are structured into seven main meta-model packages. Above these packages SPEM defines three Compliance Points (CP) which are: SPEM Complete CP, SPEM Process with Behavior and Content CP and SPEM Method Content CP. The scope of our solution is covered with Compliance Point "SPEM Complete". The reason of this compliance point is because we need to use all SPEM elements, where the most important are defined within the Method Content package, the Process with Method package and the Method Plugin package. The Method Content metamodel package provides the concepts for SPEM users and organizations to build up a development knowledge base that is independent of any specific processes and development projects. The Method Content elements are the core elements of every method such as Roles, Tasks, and Work Product Definitions. The second necessary metamodel package that we need is the Process with Method metamodel package. Process with Methods defines the structured work definitions that need to be performed to develop a system, e.g., by performing a project that follows the process. Such structured work definitions delineate the work to be performed along a timeline or lifecycle and organize it in so-called breakdown structures. Finally, the third necessary package is the Method Plugin metamodel package. The Method Plugin allows extensibility and variability mechanisms for Method Content and Process specification. It provides more flexibility in defining different variants of method content and processes by allowing content and process fragments to be plugged-in on demand, thus creating tailored or specialized content only when it is required and which can be maintained as separate units worked on by distributed teams.

2.1 Moving SPEM into the Semantic Web

In order to enable use of SPEM in the Semantic Web technical space, we make use of the fact that OWL, ODM and SPEM are serialized in XML format (OMG 2007). The mapping between OWL and ODM is expressed in ODM that contains OWL Metamodel (Djurić 2006). The OWL Metamodel is a MOF2 compliant metamodel that allows a user to specify ontologies using the terminology and underlying model theoretic semantics of OWL. Thus only a mapping between SPEM and OWL has to be created. Since the hallmark work (Gašević 2009) proposes the transformation of a MDA standard to the Semantic Web technical space with a mapping between UML Ontology Profile and an arbitrary UML Profile, we have either used this principle, thus we have created a mapping between the Ontology UML Profile and the SPEM UML Profile as it is shown in Figure 1. For more detailed and comprehensive description about the SPEM transformation to the Semantic Web technical space and its utilizations, a reader may refer to (Líška 2010b).

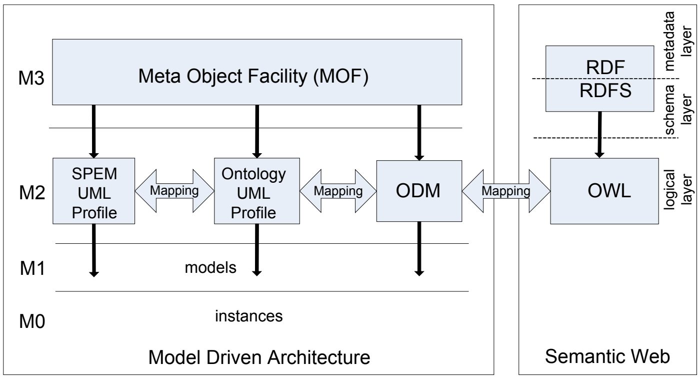


Fig. 1. Mapping between SPEM and OWL

2.2 Ontology Based Software Project Enactment

The main idea of our approach consists of two major steps. The first step is OWL DL consistency verification between two method plugin ontologies, which represent different methods and processes of a company and its supplier. In addition to be assured that both method plugins are specified correctly with SPEM, it is necessary that either the SPEM ontology is included in the verification. Moreover, since the scope of SPEM is purposely limited to the minimal elements necessary to define any software and systems development process, SPEM does not provide concepts such as Iteration, Phase etc. Therefore an ontology of the SPEM Base Plugin has to be included to the reasoning as well. However, to enable OWL DL consistency verification between set of ontologies, a mapping between their elements has to be created first (Shvaiko 2007), because some of them are usually related.

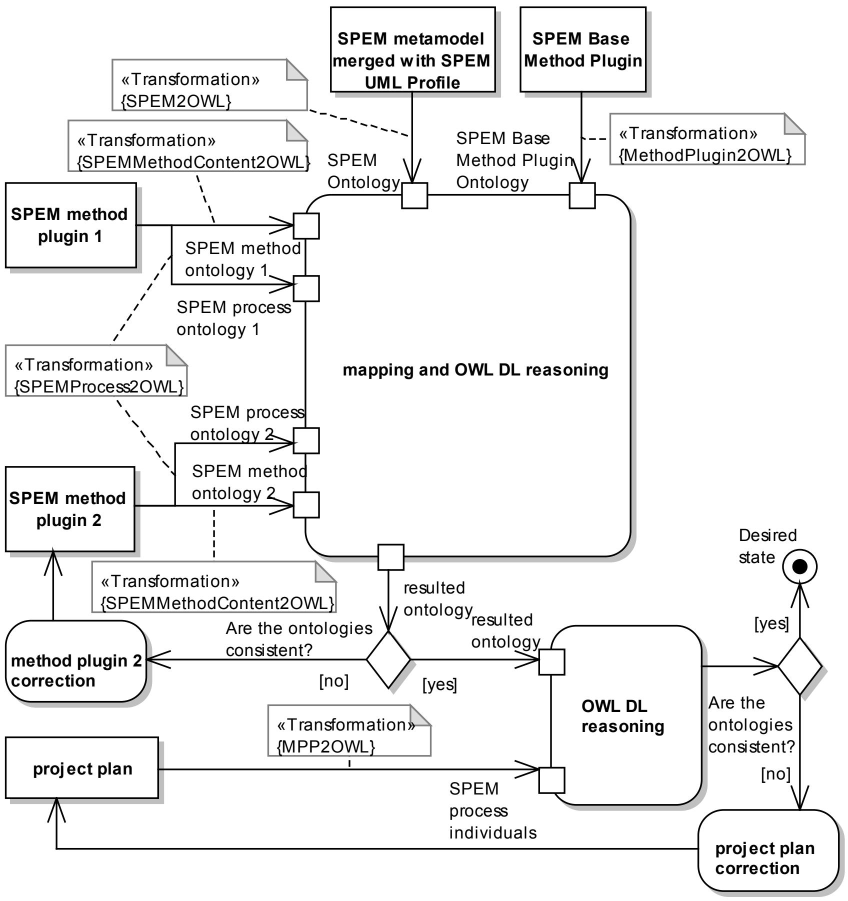

SPEM metamodel merged with SPEM UML Profile
SPEM Base Method Plugin
«Transformation» {SPEM2OWL}
«Transformation» {SPEMMethodContent2OWL}
«Transformation» {MethodPlugin2OWL}
SPEM Ontology
SPEM Base Method Plugin Ontology
SPEM method plugin 1
SPEM method ontology 1
SPEM process ontology 1
«Transformation» {SPEMProcess2OWL}
mapping and OWL DL reasoning
SPEM process ontology 2
SPEM method ontology 2
SPEM method plugin 2
«Transformation» {SPEMMethodContent2OWL}
Desired state
resulted ontology
resulted ontology
Are the ontologies consistent?
method plugin 2 correction
[no]
[yes]
OWL DL reasoning
[yes]
Are the ontologies consistent?
[no]
«Transformation» {MPP2OWL}
project plan
SPEM process individuals
project plan correction

Fig. 2. Approach to Software Project Enactment with a Supplier

The second step of our approach extends the ontology based project plan verification (Líška 2010) with a set of method plugin ontologies. Once the consistency between these method plugin ontologies is established, it is necessary also to validate it either with concrete project plan that covers mutual development between company and its supplier. Likewise as SPEM supports a process enactment with a project planning system with instantiation relation, we use information defined within a project plan as individuals of a SPEM process ontology; hence the OWL DL consistency reasoning can be executed. For the sake of clarity, the approach is depicted in Figure 2.

The figure shows both mentioned steps, the mapping and OWL reasoning with two method plugin ontologies and the OWL DL reasoning with a project plan. Since a SPEM method plugin consists of a SPEM method content and a SPEM process, we have created transformations *SPEMMethodContent2OWL* and *SPEMProcess2OWL* for a SPEM method plugin transformation to the

Semantic Web technical space. The former transforms a SPEM method content model to a SPEM method ontology, and the latter transforms a SPEM process model to a SPEM process ontology. Therefore, the Method Plugin 1 (i.e. company's method plugin) is transformed to the SPEM Method Ontology 1 and the SPEM Process Ontology 1; whereas the Method Plugin 2 (i.e. supplier's method plugin) is transformed to the SPEM Method Ontology 2 and the SPEM Process Ontology 2. In like manner, the project plan is transformed with XSL transformation *MPP2OWL* to the individuals of the SPEM Process Ontology 1. Since we have created mapping between the Method Plugin 1 and Method Plugin 2, an OWL DL reasoner will realize the individuals of the project plan either in the SPEM Process Ontology 2. The following paragraphs present the usage scenarios which our approach provides.

– Scenario 1. **Verification of a set of SPEM methods with ontology.** This scenario can be used for verification, whether at least two different SPEM method contents are consistent. Since it is necessary to manage a lot of differences such as different tasks, software work products, guidelines, roles etc., this scenario can be used to reveal and to remove such differences that are inconsistent.
– Scenario 2. **Verification of a set of SPEM processes with ontology.** This scenario is similar than the previous, but this time processes of a software development are the subject for the OWL DL verification. The scenario is used to verify, whether at least two different SPEM processes are consistent
– Scenario 3. **Project plan generation of a set of method plugins with ontology.** This scenario can be executed when a project manager wants to create a project plan that is based at least on two method plugins. First it is necessary to select the desired method contents and a processes from the set of method plugins a transform them to the ontologies. Then the scenarios 1 and 2 can be used to reveal inconsistencies. If the ontologies are consistent, then the reversed XSL transformation *OWL2SPEMProcess* is executed for the project plan generation.
– Scenario 4. **Project plan verification of a set of method plugins with ontology.** This scenario can be executed when a project manager wants to verify a project plan with a set of method plugins. When he makes changes to his project plan or assigns resources to the tasks, he can ensure that the changes he made still preserve the consistency between his project and a SPEM process.

To be more precise, we give the formally defined conditions that cover the mentioned utilization scenarios. Since the scenario 3 consists of the scenario 1 and 2 we only present the formal specification of scenarios 1, 2 and 4. Scenario 1 is covered with Formula 4, scenario 2 with Formula 5 and scenario 4 with Formulas 3 and 6. First we define the two method plugins and the Project Planning Knowledge:

$$\begin{aligned} SPEMMethodPlugin1Ontology = {} & SPEMMethodOntology1 \sqcup \\ & SPEMProcessOntology1 \end{aligned} \tag{1}$$

$$\begin{aligned} SPEMMethodPlugin2Ontology = {} & SPEMMethodOntology2 \sqcup \\ & SPEMProcessOntology2 \end{aligned} \tag{2}$$

$$\begin{aligned} ProjectPlanningKnowledge = {} & SPEMOntology \sqcup \\ & SPEMBasePluginOntology \sqcup \\ & SPEMMethodPlugin1Ontology \sqcup \\ & SPEMMethodPlugin2Ontology \end{aligned} \tag{3}$$

Then we say that the two SPEM method contents are consistent if

$$\begin{aligned} SPEMOntology \vdash SPEMBasePluginOntology \vdash \\ SPEMMethodOntology1 \vdash \\ SPEMMethodOntology2 \end{aligned} \tag{4}$$

and the two SPEM processes are consistent if

$$\begin{aligned} SPEMOntology \vdash SPEMBasePluginOntology \vdash \\ SPEMMethodPlugin1Ontology \vdash \\ SPEMMethodPlugin2Ontology \end{aligned} \tag{5}$$

Finally, the Project Planning Knowledge is satisfied in a project plan if

$$ProjectPlan \vDash ProjectPlanningKnowledge \tag{6}$$

3 Example with Formal Model

This section presents an example of our ontology based approach to software project enactment with a supplier in Description Logic (Baader 2004). First, we specify a Method Content 1 that represents a software requirements method and a Process 1 that defines a process for the Method Content 1. The Method Content 1 contains two separated Role Definitions that are the Business Analyst and Requirements Specifier. The former performs the Task Definition "Create Business Analysis" and the later "Create Requirements". For the sake of simplicity, the formal models are focused to the Role Definition "Requirements Specifier". However, the Method Content 1 and Method Content 2 constitute the Method Plugin 1 that is a company's method plugin in our case.

Secondly, we present similar software requirements method, a Method Content 2 that contains only one Role Definition "Analyst", which performs the Task Definitions "Create BPM" (i.e. a Business Process Model) and "Create SRS" (i.e. a Software Requirements Specification). For the sake of variability we do not use any process for the Method Content 2, thus the Method Plugin 2 (i.e. suppliers method plugin in our case) consists only of the Method Content 2.

Consequently, we present the usability of the first usage scenario that is the verification of the set of SPEM method contents with ontology. Formula 9 presents the case when the mentioned method contents are consistent, whereas Formula 10 presents their inconsistency. Then it is possible to use third usage scenario that is a project plan generation with the set of method plugins with ontology. The generated project plan is depicted in Figure 5. Finally we present usability of the fourth usage scenario that is the project plan verification with a set of method plugins with ontology. Formula 11 illustrates a case when the project planning knowledge is not satisfied in the project plan, whereas Formula 12 present the opposite situation. So, Figure 3 shows the Method Content 1.

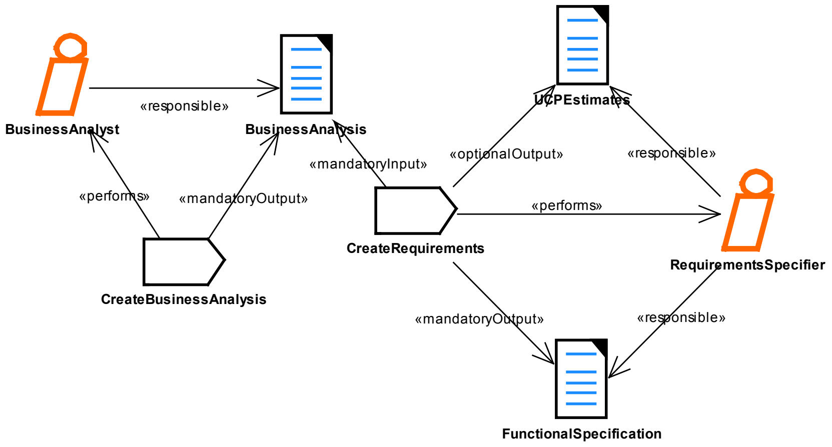


Fig. 3. Method Content 1

An excerpt of its formal model is as follows:

MethodContent1Ontology=
RequirementsSpecifier $\sqsubseteq$ RoleDefinition $\sqcap$
responsible.(FunctionalSpecification $\sqcup$ UCPEstimates),
FunctionalSpecification $\sqsubseteq$ WorkProductDefinition,
UCPEstimates $\sqsubseteq$ WorkProductDefinition,
CreateRequirements $\sqsubseteq$ TaskDefinition $\sqcap$
$\forall$ mandatoryOutput.FunctionalSpecification $\sqcap$
$\forall$ optionalOutput.UCPEstimates $\sqcap$
$\forall$ performs.RequirementsSpecifier,
BusinessAnalyst $\sqcap$ RequirementsSpecifier $\equiv \emptyset$.

Consequently we give an excerpt of a formal model of Process1 as follows:

ProcessOntology=
RequirementsProcess1 $\sqsubseteq$ Process,
PhaseInception $\sqsubseteq$ Phase,
Iteration1 $\sqsubseteq$ Iteration,
CreateRequirementsSpecification $\sqsubseteq$ Activity,
RequirementsSpecifierI1 $\sqsubseteq$ RoleUse $\sqcap$
responsible.FunctionalSpecificationV1,

CreateRequirements $\sqsubseteq$ TaskUse $\sqcap$
$\forall$ mandatoryOutput.FunctionalSpecificationV1 $\sqcap$
$\forall$ performs.RequirementsSpecifierI1.

In like manner, Figure 4 illustrates the second (i.e. supplier's) Method Content 2 "Software Requirements".

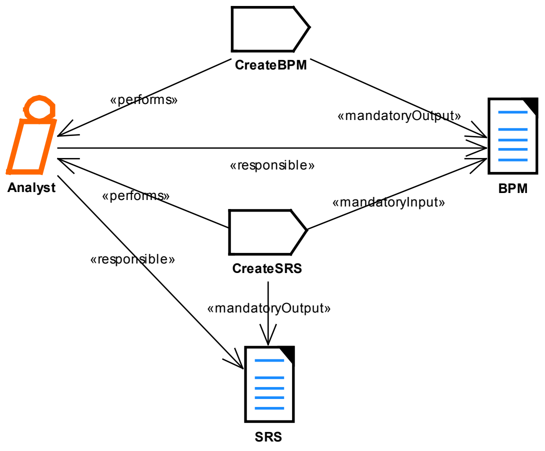


Fig. 4. Method Content 2

An excerpt of its formal model is as follows:

MethodContent2Ontology=
Analyst $\sqsubseteq$ RoleDefinition $\sqcap$
$\forall$ responsible.(BPM $\sqcup$ SRS),
CreateSRS $\sqsubseteq$ TaskDefinition $\sqcap$
$\forall$ mandatoryInput.BPM $\sqcap$
$\forall$ mandatoryOutput.SRS $\sqcap$
$\forall$ performs.Analyst.

If we state that the mappings between the Method Content 1 and Method Content 2 are

$$Analyst \equiv BusinessAnalyst \tag{7}$$

$$Analyst \equiv RequirementsSpecifier \tag{8}$$

then it is true that

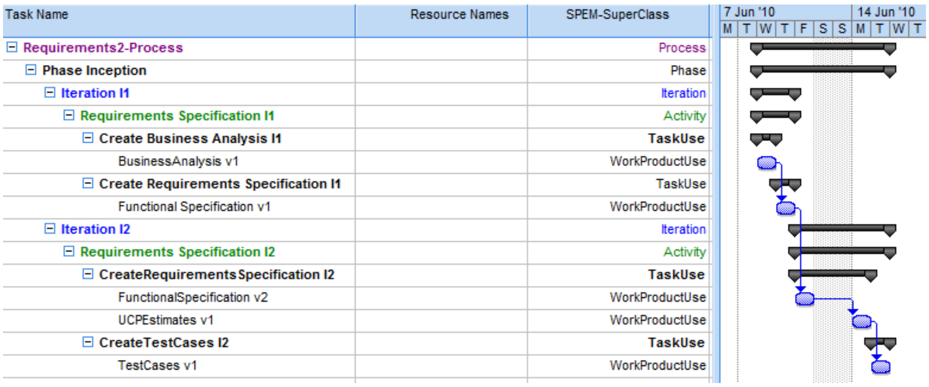
$$MethodContent1 \nvdash MethodContent2 \tag{9}$$

Formula 9 means that the Method Content 1 is not consistent with the Method Content 2, thus it is not possible to establish enactment of these methods in a software project. The reason of the inconsistency is because the Method Content

1 does not permit that the same Role Definition ”Analyst” performs both Task Definitions ”Create Requirements Specification” and ”Create Business Analysis”. To avoid this inconsistency, we have to remove asserted axiom which states that the Business Analyst and Requirement Specifier cannot be the same person, for example. After this is done, then it is true that

$$MethodContent1 \vdash MethodContent2 \tag{10}$$

Now, since we have established consistency between the Method Content 1 and Method Content 2, we can generate a project plan from the Process 1. The generated project plan is depicted in Figure 5.



Task Name	Resource Names	SPEM-SuperClass
⊟ Requirements2-Process		Process
⊟ Phase Inception		Phase
⊟ Iteration I1		Iteration
⊟ Requirements Specification I1		Activity
⊟ Create Business Analysis I1		TaskUse
BusinessAnalysis v1		WorkProductUse
⊟ Create Requirements Specification I1		TaskUse
Functional Specification v1		WorkProductUse
⊟ Iteration I2		Iteration
⊟ Requirements Specification I2		Activity
⊟ CreateRequirementsSpecification I2		TaskUse
FunctionalSpecification v2		WorkProductUse
UCPEstimates v1		WorkProductUse
⊟ CreateTestCases I2		TaskUse
TestCases v1		WorkProductUse

Fig. 5. The project plan generated from the Process 1

The figure shows that the tasks defined in the project plan up to now have not assigned resources. We set these assignments in the project plan environment and then we generate the project plan ontology. An excerpt of its formal model is as follows:

ProjectPlan1Ontology=
BusinessAnalyst(”BusinessAnalyst1”),
RequirementsSpecifier(”RequirementsSpecifier1”),
TestAnalyst(”TestAnalyst1”),
performs(”CreateBusinessAnalysis1”,”BusinessAnalyst1),
performs(”CreateRequirementsSpecification1”, ”BusinessAnalyst1”),
performs(”CreateRequirementsSpecification1”, ”TestAnalyst1”).

It can be proved that

$$ProjectPlan1Ontology \nvDash (MethodPlugin1 \sqcup MethodPlugin2) \tag{11}$$

The reason of this inconsistency is because we have intentionally stated that the Task Use” CreateRequirementsSpecification1” performs the Role Use ”Business-Analyst”, what is in the contradiction with the SPEM Process 1 ontology, where it is stated that the performer is the Requirements Specifier. Hence, when this is corrected, then it is true that

$$ProjectPlan1Ontology \vDash (MethodPlugin1 \sqcup MethodPlugin2) \tag{12}$$

4 Conclusions

We have presented our approach to software project enactment with a supplier. We shown the new approach how ontologies can support a real software project in the context of the selected SWEBOK knowledge areas (i.e. process definition, project planning and project enactment). Likewise as another MDA' standard UML has brought a new generation into specification, documentation and realization of information systems, it is very probable that SPEM will play the same role in the domain of software process engineering. Hence we believe that our presented approach is at least in the right direction, since we aim to improve the SPEM capabilities with the OWL DL reasoning. However, when we compare our work with the closest approach (Rodríguez 2009) we conclude that we have created the more accurate SPEM ontology architecture with respect to the actual SPEM metamodel. Likewise as SPEM provides concepts for a method content, a process or a method plugin, our approach is based on a method content ontology, a process ontology and either method plugin ontology. Moreover, the work is not concerned with method contents or processes integrations, hence neither with the software project enactment with a supplier. However, we cannot to say that our approach of project plan verification that was either used in this approach is more accurate (with respect to the SPEM metamodel), rather is more comprehensive. If we focus only on the substantive part of project plan verification, the work uses instantiation relation between a project plan and a SPEM process either, thus it conforms to the project plan enactment as it SPEM defines. Additionally, the work proposes use of SWRL that add rules to the reasoning process, thus we conclude, that it proposes a project plan reasoning with advanced expressiveness.

Acknowledgments. "This work was partially supported by the Scientific Grant Agency of Slovak Republic, grant No. VG1/508/09."

References

Baader, F., Horrocks, I., Saatler, U.: Description Logics. In: Handbook on Ontologies, International Handbooks on Information Systems, pp. 3–28. Springer, Heidelberg (2004)

Brickley, D., Guha, R.V., McBride, B.: RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation (2004)

Calero, C., Ruiz, F., Piattini, M.: Ontologies for Software Engineering and Software Technology. Springer, Heidelberg (2006)

Cranefield, S.: Networked Knowledge Representation and Exchange using UML and RDF. Journal of Digital Information 1(8) (2001)

Djurić, D.: MDA-based ontology infrastructure. Computer Science and Information Systems 1(1), 91–116 (2006)

Falbo, R.A., Guizzardi, G., Duarte, K.C.: An Ontological Approach to Domain Engineering. In: Proceedings of 14th International Conference on Software Engineering and Knowledge Engineering (SEKE), Ischia, Italy, pp. 351–358 (1992)

Frankel, D.S.: Model Driven Architecture. In: Applying MDA to Enterprise Computing. Willey, USA (2003)

Gašević, D., Djurić, D., Devedžić, V.: Bridging MDA and OWL Ontologies. Journal of Web Engineering 4(2), 119–134 (2005)

Gašević, D., Djurić, D., Devedžić, V.: MDA and Ontology Development. Springer, Heidelberg (2006)

Gašević, D., Djurić, D., Devedžić, V.: Model Driven Engineering and Ontology Development, 2nd edn. Springer, Berlin (2009)

Happel, H.J., Seedorf, S.: Applications of ontologies in software engineering. In: International Workshop on Semantic Web Enabled Software Engineering SWESE 2006, Athens, USA (2006)

Hart, L., Emery, P., Colomb, B., Raymond, K., Taraporewalla, S., Chang, D., Ye, Y., Kendall, E., Dutra, M.: OWL Full and UML 2.0 Compared. OMG TFC Report (2004)

Hilera, J.R., Sánchez-Alonso, S., García, E., Del Molino, C.J.: OntoGLOSE: A Lightweight Software Engineering Ontology. In: 1st Workshop on Ontology, Conceptualizations and Epistemology for Software and Systems Engineering (ONTOSE). Alcalá de Henares, Spain (2005)

Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, T., Grosof, B., Dean, M.: SWRL: A Semantic Web Rule Language. Combining OWL and RuleML. W3C Member Submission (2004)

IEEE Computer Society: IEEE Std 610.12-1990(R2002). IEEE Standard Glossary of Software Engineering Terminology. IEEE, New York, USA (2002)

IEEE Computer Society: Software Engineering Body of Knowledge - SWEBOK. Angela Burgess, EUA (2004)

Kruchten, P.: The Rational Unified Process: An Introduction, 3rd edn. Addison-Wesley Professional, Reading (2003)

Kurtev, I., Bézivin, J., Aksit, M.: Technological spaces: An initial appraisal. In: Meersman, R., Tari, Z., et al. (eds.) CoopIS 2002, DOA 2002, and ODBASE 2002. LNCS, vol. 2519. Springer, Heidelberg (2002)

Larburu, I.U., Pikatza, J.M., Sobrado, F.J., García, J.J., López, D.: Hacia la implementación de una herramienta de soporte al proceso de desarrollo de software. In: Workshop in Artifificial Intelligence Applications to Engineering (AIAI), San Sebastián, Spain (2003)

Líška, M.: Extending and Utilizing the Software and Systems Process Engineering Metamodel with Ontology. PhD Thesis, ID:FIIT-3094-4984. Slovak Technical University in Bratislava (2010)

Líška, M., Návrat, P.: An Approach to Project Planning Employing Software and Systems Engineering Meta-Model Represented by an Ontology. Computer Science and Information Systems Journal (conditional acceptance with minor revision) (2010)

Líška, M.: An Approach of Ontology Oriented SPEM Models Validation. In: Proceedings of the First International Workshop on Future Trends of Model-Driven Development (FTMDD), Conjuction with 11th International Conference on Enterprise Information Systems, pp. 40–43. INSTICC Press, Milan (2009)

Mcguinness, D.L., Harmelen, F.: OWL Web Ontology Language Overview. W3C Recommendation (2004)

Mendes, O., Abran, A.: Issues in the development of an ontology for an emerging engineering discipline. In: First Workshop on Ontology, Conceptualizations and Epistemology for Software and Systems Engineering (ONTOSE). Alcalá de Henares, Spain (2005)

Object Management Group: Meta Object Facility (MOF) 2.0 Core Specification. Object Management Group, USA (2006)

Object Management Group: MOF 2.0 / XMI Mapping Specification, v2.1.1. Object Management Group, USA (2007)

Object Management Group: Ontology Definition Meta-Model 1.0. Object Management Group, USA (2009)

Object Management Group: UML 2.2 Infrastructure Specification. Object Management Group, USA (2009)

Object Management Group: UML 2.2 Superstructure Specification. Object Management Group, USA (2009)

Object Management Group: Software and Systems Process Engineering Meta-Model 2.0. Object Management Group, USA (2008)

Pan, J., Horrocks, I.: Metamodeling Architecture of Web Ontology Languages. In: Proceedings of the First Semantic Web Working Symposium, Stanford, USA, pp. 131–149 (2001)

Rodríguez, D., Sicilia, M.A.: Defining SPEM 2 Process Constraints with Semantic Rules Using SWRL. In: Proceedings of the Third International Workshop on Ontology, Conceptualization and Epistemology for Information Systems, Software Engineering and Service Science Held in Conjunction with CAiSE 2009 Conference, Amsterdam, The Netherlands, pp. 95–104 (2009)

Shvaiko, P., Euzenat, J.: Ontology Matching. Springer, Heidelberg (2007)

Schwaber, K., Beedle, M.: Agile Software Development with SCRUM. Prentice-Hall, Englewood Cliffs (2002)

Sicilia, M.A., Cuadrado, J.J., García, E., Rodríguez, D., Hilera, J.R.: The evaluation of ontological representation of the SWEBOK as a revision tool. In: 29th Annual International Computer Software and Application Conference (COMPSAC), Edinburgh, UK (2005)

Smith, M.K., Welty, C., McGuinness, D.L.: OWL Web Ontology Language Guide. W3C Recommendation (2004)

Steinberg, D., Budinsky, F., Paternostro, M., Merks, E.: EMF: Eclipse Modeling Framework, 2nd edn. Addison-Wesley Longman, Amsterdam (2009)

Zualkernan, I.A.: An Ontology-Driven Approach for Generating Assessments for the Scrum Software Process. In: Proceedings of the seventh SoMeT, pp. 190–205. IOS Press, The Netherlands (2008)

Determining Objects within Isochrones in Spatial Network Databases

Sarunas Marciuska and Johann Gamper

Free University of Bolzano-Bozen, Italy
{gamper,marciuska}@inf.unibz.it

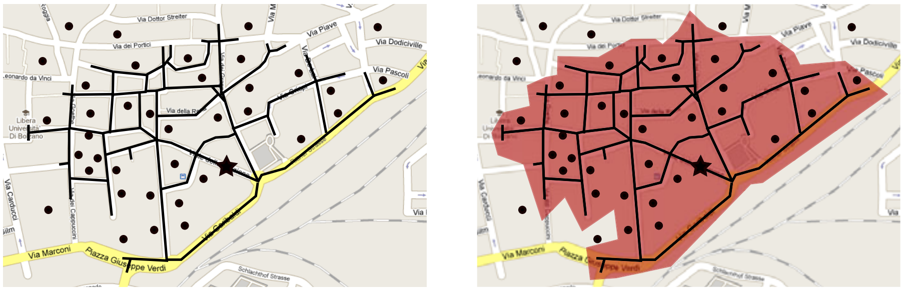
Abstract. Isochrones are generally defined as the set of all space points from which a query point can be reached in a given timespan, and they are used in urban planning to conduct reachability and coverage analyzes in a city. In a spatial network representing the street network, an isochrone is represented as a subgraph of the street network. Such a network representation is not always sufficient to determine all objects within an isochrone, since objects are not only *on* the network but might be in the immediate vicinity of links (e.g., houses along a street). Thus, the spatial area covered by an isochrone needs to be considered.

In this paper we present two algorithms for determining all objects that are within an isochrone. The main idea is to first transform an isochrone network into an isochrone area, which is then intersected with the objects. The first approach constructs a spatial buffer around each edge in the isochrone network, yielding an area that might contain holes. The second approach creates a single area that is delimited by a polygon composed of the outermost edges of the isochrone network. In an empirical evaluation using real-world data we compare the two solutions with a precise yet expensive baseline algorithm. The results demonstrate the efficiency and high accuracy of our solutions.

1 Introduction

Urban planning has to deal with tasks such as to analyze the reachability of strategic objects in a city and to place these objects in optimal positions. For example, what is the best place to build a metro station or a school such that a large number of people can reach that place in comfortable times?

Isochrones, which are defined as the set of all space points from which a query point can be reached in a given timespan, are used as an instrument to perform such analyses. By joining an isochrone with the inhabitants database the number of citizens living in a certain distance from a query point can be determined. Figure 1(a) shows the 5 minutes isochrone for a single query location (star) in the city of Bozen-Bolzano. The isochrone is represented by the street segments in bold and covers all points in the street network from where the query point can be reached in 5 minutes, assuming a walking speed of 1.6 m/s and considering walking as the only mode of transportation. In general, the computation of isochrones needs to consider multiple modes of transportation, e.g., walking, bus, train, metro, etc. An isochrone is then a possibly disconnected set of space points: a large area around the query point and smaller areas around bus/metro/train stops, from where the query point can be reached by a combination of walking and using public transportation. For the sake of simplicity, in this paper we consider mainly isochrones in a pedestrian network (walking mode only). The proposed solutions can easily be extended for multiple modes of transportation, as we briefly discuss in Sec. 4.



(a) Isochrone Network (b) Isochrone Area

Fig. 1. Isochrone Representation as Network and Area

The representation of isochrones as a subgraph of the street network is not always sufficient. A representation as an area, such as illustrated in Fig. 1(b), is often desirable for several reasons. First, the objects within an isochrone we are looking for might not lie exactly on the network but in the immediate vicinity of links, e.g., houses in a street have usually a distance of up to 50 meters or more from the street. In Fig. 1 such objects are represented as dots. With a representation as an area, it is straightforward to determine all (static or dynamic) objects within an isochrone. Second, for human users isochrones are usually visualized as an area rather than as a subgraph. Therefore, we aim at transforming an isochrone network representation into an isochrone area representation that covers the (immediate) vicinity of the isochrone network.

The computation of an isochrone area from an isochrone network is similar to the computation of a footprint for a set of points. The most common methods for this are concave hull [10] and alpha shapes [4]. Since these methods compute an area from a set of 2D points and we have a set of 2D links, they cannot directly be applied. By transforming the links into a set of points, we loose the edge information which might result in large errors, as illustrated in Fig. 2. The isochrone network in Fig. 2(a) is transformed into a set of points in Fig. 2(b). Figure 2(c) shows the area that is obtained with the alpha shapes or concave hull method; the large area indicated by the letter “A” is missing. While a parameter allows to control the computation of the area, it is generally impossible to find the right parameter to obtain the correct area, and the problem remains that with the transformation into a set of points relevant pieces of spatial information are lost.

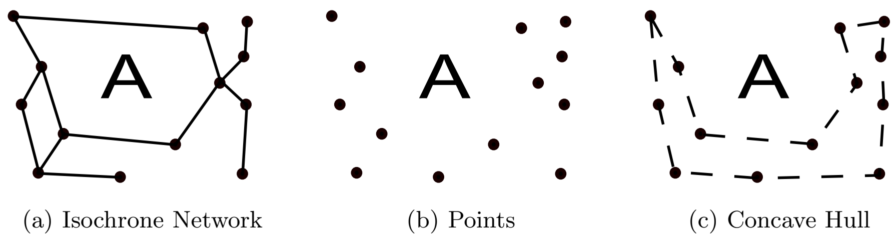


Fig. 2. Concave Hull (and Alpha Shapes) Method

In this paper we present two different solutions to transform an isochrone network into an isochrone area. The *link-based approach* constructs a buffer of a user-specific size around each individual link of the isochrone network, yielding an area that possibly contains holes. This solution exploits existing spatial database functionalities. The *surface-based approach* computes first a polygon that covers the isochrone network and then creates a buffer around this polygon. The obtained area doesn't contain holes. To determine all objects within an isochrone, the constructed area is intersected with the relation that stores the objects. We empirically evaluate the two solutions using real-world data and determining all objects within an isochrone. We measure the quality of each solution by comparing it with a baseline solution that is precise but expensive in terms of runtime. As quality estimators we use recall, precision, and f-measure. The experiments show that the quality of the surface-based approach is higher for buffers smaller than approximately 60 meters. For a larger buffer size both approaches give similar results, since the area constructed by the link-based approach contains less holes and becomes more similar to the area constructed by the surface-based approach. The surface-based approach is faster than link-based approach, though both approaches scale almost linearly with the size of the isochrone.

The rest of the paper is organized as follows. In Section 2 we discuss related work. Sections 3 and 4 present the two different approaches for the computation of an isochrone area. Section 5 presents the experimental results, and Section 6 draws conclusions and points to future work.

2 Related Work

Isochrones are first introduced in [1,7] as a new query type in spatial network databases, which is used by city planners as an instrument to analyze coverage and reachability queries in a city. The work in [1,7] computes isochrones for bimodal networks, consisting of a pedestrian network and one or more bus networks. Since isochrones are represented as a subgraph of the pedestrian network, only objects that lie exactly on the network edges can be determined. In this paper we extend this work to represent isochrones as areas and to determine all objects that lie within this area.

Among the various queries in spatial network databases, range queries are closest to isochrones. A range query determines all objects (of a specific type) that are within a specific distance from a query point. The main difference is that an isochrone represents an area (containing all space points) from where a query point is reachable in a given timespan, while a range query returns all objects within a given distance. An isochrone can be intersected with any type of objects without recomputing it from scratch, and it can also be graphically visualized for a human user.

Range queries for spatial network databases are first introduced in [12], where the Range Euclidean Restriction (RER) algorithm and the Range Network Expansion (RNE) algorithm are presented. Deng et al. [2] improve over the work in [12] by performing less network distance calculations and therefore accessing less network data. Mouratidis et al. [11] present a solution for continuous nearest neighbor monitoring in road networks, where the query points and the data objects move frequently and arbitrarily in the network. All these frameworks for range queries and nearest neighbor queries assume that the objects lie exactly on the network links. Isochrones, in particular the isochrone areas as constructed in this paper can also catch objects that are in a (user-specified) immediate vicinity of the links.

The work which is closest to our work is the continuous intersection join presented in [13]. It proposes a solution to determine all objects that can be reached from a moving query point within a specified time. The main idea is to use a distance range query from the query point in order to determine the objects that can be reached from there. However, the range query uses the Euclidean distance resulting in a circular area around the query point, which is intersected with the objects. Isochrones use the network distance, and the main challenge is to construct a minimal area around the isochrone network which represents all space points within the isochrone.

The computation of an isochrone area from an isochrone network is similar to the computation of a convex or concave hull for a finite set of 2D points. Two main algorithms are known for the concave hull: Jarvis March [9] and Graham scan [8]. The main idea of the Jarvis March approach [9] is to include a point in the convex hull that has the smallest polar angle with respect to a previous point. As the initial point, the left-most point among all points is taken. The algorithm runs in $O(nh)$ time, where n is the number of points in the data set and h is a number of points on the convex hull. The Graham scan approach [8] works in three steps. First, the point with the smallest y-coordinate is chosen. Second, the remaining points are increasingly sorted according to the x-coordinate. Finally, for each next point in a convex hull, the turn between the point and the previous two points is computed. If it is a right turn, the link from the second to the last point is removed. If a left turn occurs, the last point is included into the convex hull, and the next point is taken from the sorted array. The algorithm runs in $O(n \log n)$ time for a finite set of n points.

In general, the shape of an isochrone area is closer to a concave hull than to the convex hull. Different from the convex hull, there is no unique concave hull. An algorithm for the computation of concave hulls for a set of 2D points is presented in [10]. The algorithm is based on the k-nearest neighbors. Depending on the choice of k, different concave hulls are generated. With a higher number of k, the shape becomes smoother. With $k = n$, the concave hull coincides with the convex hull. It is difficult to determine the right value of k to get a good shape for the isochrone area.

A similar problem of finding footprints for a set of $2D$ points is discussed in [6] and [3,4,5]. These approaches are based on so-called alpha shapes. The main idea of the alpha shape algorithm is to draw circles with a radius of $1/alpha$ such that they touch at least two points and none of the other points is inside those circles. All points that touch a circle are selected and connected. If the radius is big enough, the result is the convex hull. If the radius is too small, the result is the set of all points without any connections.

Since isochrone networks are represented as 2D links, the above approaches for the computation of convex/concave hulls and alpha shapes cannot be directly applied to compute isochrone areas. If we first transform the links into points, we loose important spatial information, which might lead to significant errors in the shape of the isochrone area.

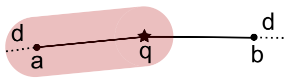
3 Link-Based Approach

In this section we describe the link-based approach, which draws a buffer around each individual link of the isochrone network and returns the union of these buffers as the isochrone area.

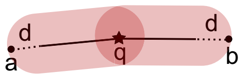
An isochrone network is represented as a graph $G = (V, E, \gamma)$, where V is a set of vertices (or nodes), $E \subseteq V \times V$ is a set of links (or edges), and γ is a function that assigns a geometry to each edge. The geometry is a polyline, $\gamma((u, v)) = \{p_1, \dots, p_n\}$, where $p_1, \dots, p_n$ are space points that are connected by lines.

To create the buffers we use Oracle's built-in function SDO_BUFFER(A,d), which creates a buffer of size d around the spatial object A. Unfortunately, by applying this function for a link, the border of the buffer is not going through the endpoints of the link, as illustrated in Figure 3(a). The isochrone consists of the nodes $V = \{q, a, b\}$ and the links $E = \{(q, a), (q, b)\}$, where a and b represent the outermost points from where the query point q is reachable in the given timespan t_{max}. Drawing a buffer of distance d around each of the two links introduces an error near the nodes a and b.

To remedy from this problem, we reduce the maximal timespan, t_{max}, of the isochrone by an amount that corresponds to the buffer size d. That is, we determine $t'_{max} = t_{max} - \frac{d}{s}$ as the new timespan for the computation of the isochrone network; s is the walking speed used for the computation of the isochrone. Using t'_{max} results in a smaller isochrone network. By constructing a buffer for each link in the reduced isochrone network, the buffers cross exactly the outermost points a and b of the original network (see Fig. 3(b)).



(a)

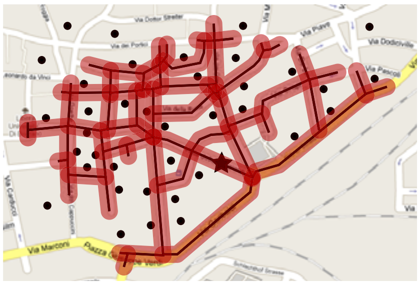


(b)

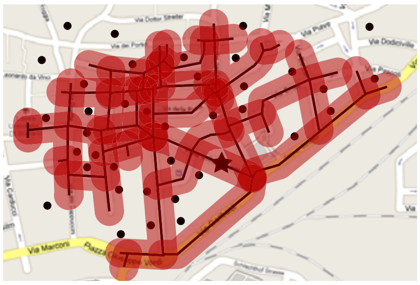
Fig. 3. Decreasing the Timespan for the Computation of the Isochrone Network

```
Algorithm: LISO$(I, d)$
Input: Isochrone $I = (V, E, \gamma)$; distance $d$;
Output: Isochrone area $B$;
$B \leftarrow \emptyset$ ;
foreach link $l \in E$ do
    $B \leftarrow B \cup \{SDO\_BUFFER(l, d)\}$;
end
return $B$;
```

Fig. 4. Link-based Approach for the Computation of an Isochrone Area



(a) Buffer Size = 30 m



(b) Buffer Size = 50 m

Fig. 5. Isochrone Area with the Link-Based Approach

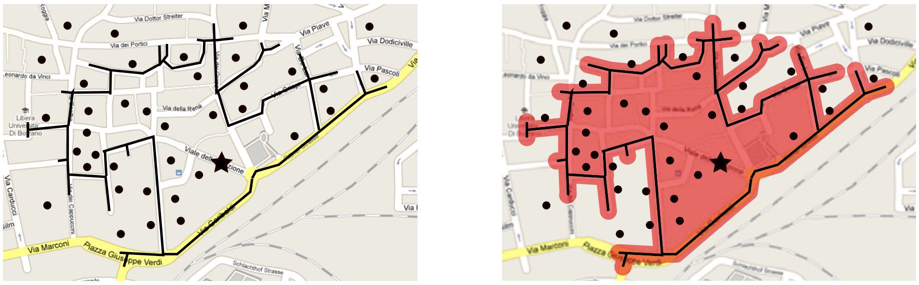
Figure 4 shows the algorithm LISO for the computation of an isochrone area using the link-based approach. The algorithm has two input parameters: an isochrone network I; a buffer size d. The algorithm iterates over all links in the isochrone network I and constructs a buffer of size d around each link. These buffers are collected in B and are returned as area representation of the isochrone I, covering all space point on the network and in the immediate vicinity from where q is reachable in the given timespan.

Figure 5(a) shows the isochrone area computed with the link-based approach, using a buffer size of 30 meters. Depending on the size of the buffer, the isochrone contains more or less holes. The isochrone in Fig. 5(b) uses a buffer size of 50 meters, resulting in less holes.

4 Surface-Based Approach

The surface-based approach computes first the minimum bounding polygon of the isochrone network, termed its surface, and draws then a buffer around the surface. The *surface* of an isochrone $I = (V, E, \gamma)$ is defined as the minimal set of links $S \subseteq E$ that form a polygon and cover all other links in E. Figure 6(a) shows the surface of the isochrone in our running example, which covers all street links of the isochrone network.

Next, we construct a buffer of a user-specified size d around the surface polygon in order to include also space points in the outer vicinity of the surface polygon. Figure 6(b) shows the isochrone area that is constructed with the surface-based approach. Obviously, the isochrone area does not contain any holes (different from the link-based approach).



(a) Surface of the Isochrone (b) Isochrone Area

Fig. 6. Isochrone Area with Surface-Based Approach

The algorithm to compute the surface of an isochrone is a generalization of Jarvis' algorithm [9] for the computation of the convex hull. The main idea is to find first the link with the left-most endpoint (i.e., smallest x-coordinate) and the smallest counter-clockwise angle with the y-axis. (Any other link which lies on the surface polygon could be used as the initial link as well). Starting from the initial link, the algorithm iteratively adds an adjacent link to the surface, which has the smallest counter-clockwise angle with the link that has been added previously. The algorithm terminates when it returns to the initial link.

Figure 7 shows the algorithm, which has two input parameters: an isochrone network I and the size d of the buffer. The algorithm returns the area representation of the isochrone I, using the surface-based approach.

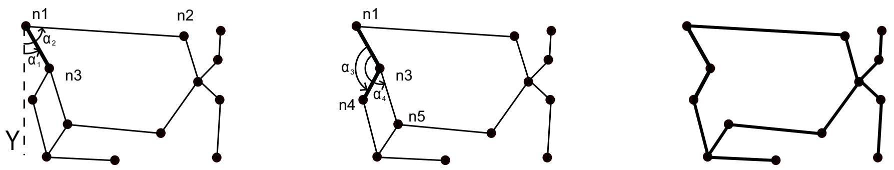
The algorithm determines first the left-most node, u_0, of the isochrone network. Then, the link through u_0 which has the smallest counter-clockwise angle with the y-axis is determined. Figure 8(a) illustrates this step. The left-most node is n_1, which has two links (n_1, n_3) and (n_1, n_2). The link (n_1, n_3) has the smaller angle, α_1, and is chosen as the initial link and added to the surface S. Next, the algorithm enters a loop, in which the surface is incrementally extended with a new link on each iteration until the initial link is encountered

```
**Algorithm:** sISO($I, d$)
**Input**: Isochrone $I = (V, E, \gamma)$; distance $d$;
**Output**: Isochrone area $B$
$u_0 \leftarrow \underset{v \in V}{\text{argmin}}\{v.x\}$;
$\alpha \leftarrow 360°$;
foreach link $(u, v) \in E$ such that $u = u_0$ do
    $\alpha' \leftarrow angle(u-(0,1), u, v)$;
    if $\alpha' < \alpha$ then
        $\alpha \leftarrow \alpha'$;
        $(u_0, v_0) \leftarrow (u, v)$;
    end
end
$S \leftarrow \{(u_0, v_0)\}$;
$(u_p, v_p) \leftarrow (u_0, v_0)$;
repeat
    $\alpha \leftarrow 360°$;
    foreach link $(u, v) \in E$ such that $u = v_p$ do
        $\alpha' \leftarrow angle(u_p, u, v)$;
        if $\alpha' < \alpha$ then
            $\alpha \leftarrow \alpha'$;
            $(u', v') \leftarrow (u, v)$;
        end
    end
    $S \leftarrow S \cup \{(u', v')\}$;
    $(u_p, v_p) \leftarrow (u', v')$;
until $(u', v') \neq (u_0, v_0)$ ;
$B \leftarrow SDO\_BUFFER(S, d)$;
return $B$
```

Fig. 7. Algorithm sISO

again. (u_p, v_p) represents the link that has been added in the previous iteration (or the initial link on the first iteration). On each iteration all links that are connected to (u_p, v_p), i.e., have v_p as source node, are considered. The link with the smallest counter-clockwise angle with (u_p, v_p) is on the surface and is added to S. This step is illustrated in Fig. 8(b). The links (n_3, n_4) and (n_3, n_5) are considered as possible extensions of the initial link (n_1, n_3). Since α_3 is smaller than α_4, the link (n_3, n_4) is added to the surface S. The loop terminates when the initial link (u_0, v_0) is encountered again. Figure 8(c) shows the completed surface. As a last step, the algorithm creates a buffer of size d around the surface of the isochrone, which is returned as a result.

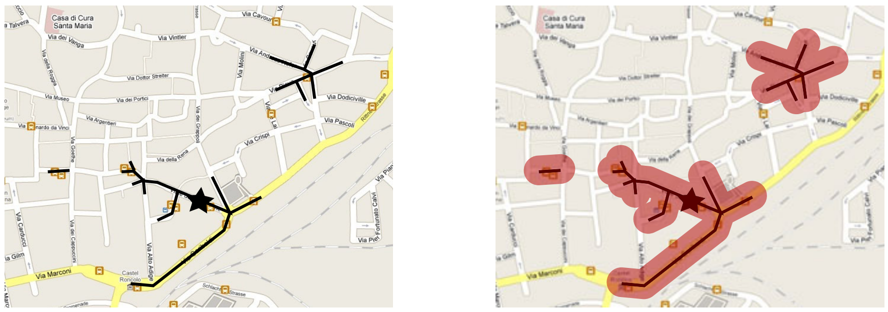
Finding the left-most link at the beginning and the next link in each iteration takes $O(n)$ time, where n is the number of nodes in the isochrone. In the worst case, the extension step iterates over all nodes (if all nodes are on the surface), yielding an overall complexity of $O(n^2)$.



(a) (b) (c)

Fig. 8. Step-Wise Computation of the Surface of an Isochrone

When multiple modes of transportation are considered, isochrones get typically disconnected. For instance, Fig. 9(a) shows an isochrone when walking in combination with buses are considered. There is a large island (area) around the query point and small islands around the reachable bus stops. While LISO correctly handles disconnected isochrones, the surface-based algorithm SISO works only for isochrones that form a connected graph. To adapt the algorithm for disconnected isochrones (as produced when multiple transportation modes are considered), a pre-processing step is required to determine the connected components (i.e., maximal connected subgraphs) of the isochrone, which can be done in linear time. Then for each connected component the algorithm SISO is called.



(a) (b)

Fig. 9. Isochrone for Multiple Modes of Transportation

5 Experimental Evaluation

In this section we present the results of an experimental evaluation of the two algorithms using real-world data.

5.1 Setup and Data

The two algorithms for the computation of an isochrone area and the intersection of the isochrone area with objects (e.g., houses) were implemented in Java on top of the Oracle Spatial DBMS. The algorithms use built-in functionalities of Oracle Spatial to construct buffers and to compute the intersection between areas and objects. To compute the initial isochrone network, which is passed as input to sISO and lISO, we use the algorithm in [7]. The spatial data, including the isochrones and the objects, are stored in the database. All experiments were run on a computer with a 2GHz CPU and 1.5 GB RAM.

For the experiments we used the street network of the city of Bolzano-Bozen, which consists of approximately 3500 links (streets segments). As objects within an isochrone we used the houses in Bolzano-Bozen (approximately 12300 houses), which are stored in a separate table.

To measure the quality of lISO and sISO we implemented a baseline approach as a reference solution, which essentially works as follows. Each house h is projected perpendicularly to the closest edge (u, v) in the pedestrian network. More specifically, it is projected to a segment (p_i, p_{i+1}) of the polyline $\gamma((u, v)) = \{p_1, \ldots, p_n\}$ that represents the edge's geometry. Assume that a house is mapped to point p on edge (u, v). Then a house is considered to be within an isochrone if its Euclidean distance to p plus the network distance from p to query point q is smaller than the maximal timespan of the isochrone. While the basline approach is slow, since it needs to determine for each house the distance to each individual segment of all edges, we use it as a reference solution to measure precision, recall, and f-measure of the surface-based and link-based solutions.

5.2 Precision, Recall, and F-Measure

Varying the Location of the Query Point. In the first experiment we use a fixed timespan of 15 min, a walking speed of 1.6 m/s, and a buffer size of 30 m, and we vary the location of the query point between locations in the center and the border of the city as well as between dense, average, and sparse areas. To measure the quality of the two approaches the f-measure is used. The result of this experiment is presented in Fig. 10 and shows that the location of the query point (and hence different densities of houses) has almost no impact on the quality. Therefore, in the remaining experiments we use a fixed query point in the city center.

Varying Buffer Size and Maximum Timespan. In the second experiment we use a fixed query point in the city center and a walking speed of 1.6 m/s, and we vary the buffer size between 10 and 120 m and the maximal timespan for the isochrone between 10 and 50 min.

Figure 11 presents the precision of sISO and lISO. For both solutions the precision depends on the size of the isochrone. The bigger the maximal timespan, the higher is the precision. Vice versa, if the size of the buffer is too large,

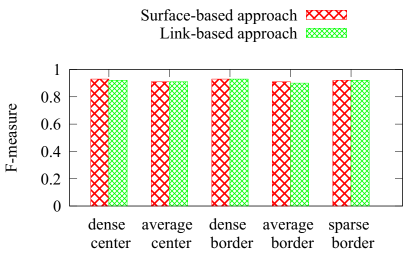
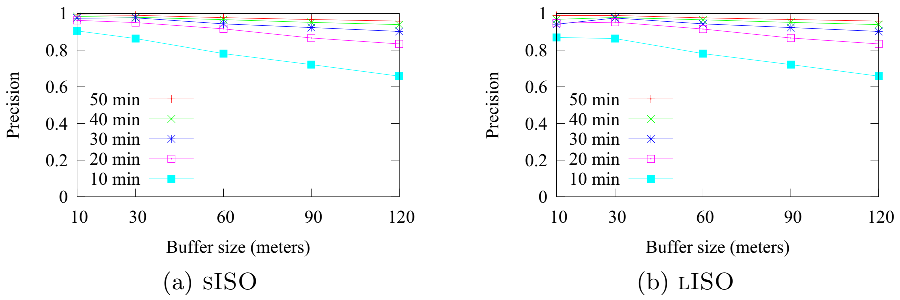


Fig. 10. F-measure for Different Query Points



(a) sISO (b) lISO

Fig. 11. Precision

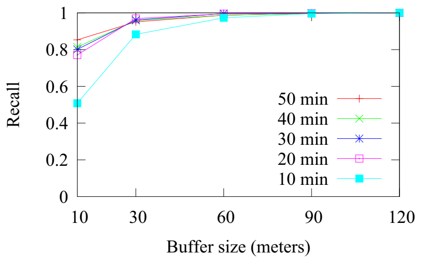
the precision is decreasing, since many false positives are included. There is no substantial difference in precision between sISO and lISO. The best precision is obtained when the buffer size is between 10 and 30 m.

Figure 12 shows the recall, which for both solutions increases with the size of the isochrone and with the size of the buffer, though the size of the buffer has less impact in lISO. While the precision is almost identical for both solutions, the surface-based approach has a significantly higher recall for small buffers up to a size of 30 m. When the buffer size is small, the link-based approach misses many objects that are located in the holes.

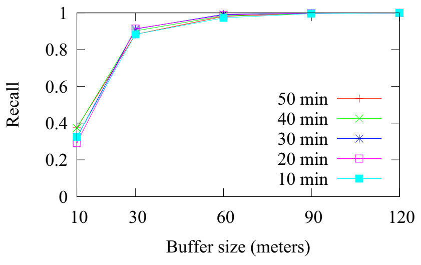
Figure 13 shows the f-measure. For a small buffer size up to approximately 60 m the surface-based approach is superior. For large buffers the difference between the two solutions disappears, since the isochrone area produced by the link-based approach becomes more and more similar to the isochrone area produced by the surface-approach. The highest f-measure for both solutions is obtained with a buffer size of approximately 60 m.

5.3 Runtime

In the last experiment we analyze the efficiency of the proposed solutions, by varying the size of isochrone, since the number of links in an isochrone is the most influencing factor for the running time. All other parameters are fixed: buffer size 60 m, walking speed 1.6 m/s, and query point in the city center.

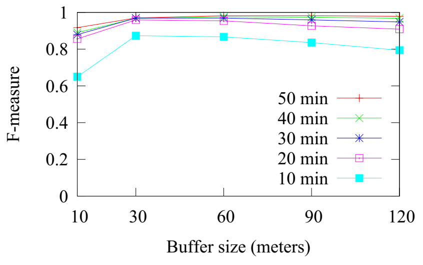


(a) sISO

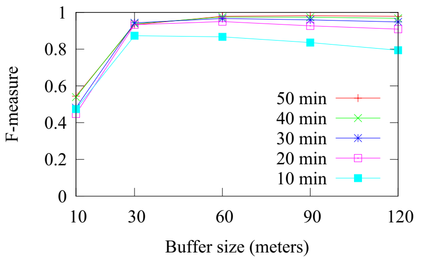


(b) lISO

Fig. 12. Recall

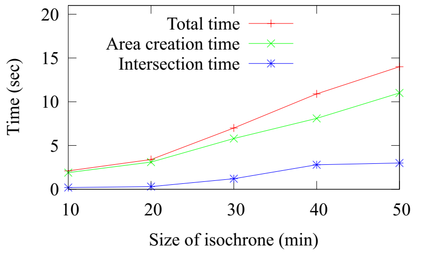


(a) sISO

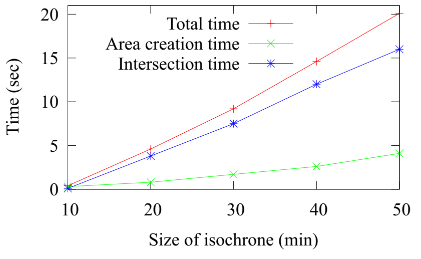


(b) lISO

Fig. 13. F-measure



(a) sISO



(b) lISO

Fig. 14. Runtime

Figure 14 shows the results of the runtime experiment, distinguishing between the time for computing the isochrone area, intersecting the area with the objects, and the total runtime. The creation of the link-based area is more efficient. However, the intersection of the isochrone area with the objects is faster in the surface-based approach, since only one large area needs to be intersected. Overall, both solutions scale almost linearly with the size of the isochrone, and the surface-based approach is faster than the link-based approach in terms of total runtime.

6 Conclusion and Future Work

In this paper we present two different solutions, termed link-based approach and surface-based approach, to determine all objects that lie within an isochrone. Both solutions first transform an isochrone network into an isochrone area and then perform an intersection with the relation that stores the objects. The link-based approach constructs a buffer around each individual link of the isochrone network. The surface-based approach computes first a polygon that covers the isochrone network and then creates a buffer around this polygon. We run experiments with real-world data to measure the quality and efficiency of the two solutions. Both approaches achieve a high quality (compared to a precise yet slow reference solution). The surface-based approach is superior for small buffers, and it is more efficient than the link-based approach.

Future work is possible in various directions. More specifically, we will conduct more extensive experiments both to study the quality of the two solutions for different types of objects and to analyze the scalability for very large isochrones.

References

1. Bauer, V., Gamper, J., Loperfido, R., Profanter, S., Putzer, S., Timko, I.: Computing isochrones in multi-modal, schedule-based transport networks (demo paper). In: ACMGIS 2008, Irvine, CA, USA, November 5-7, pp. 1–2 (2008)
2. Deng, K., Zhou, X., Shen, H.T., Sadiq, S.W., Li, X.: Instance optimal query processing in spatial networks. VLDB J. 18(3), 675–693 (2009)
3. Edelsbrunner, H.: Weighted alpha shapes. Technical Report:UIUCDCS-R-92-1760 (1992)
4. Edelsbrunner, H., Kirkpatrick, D.G., Seidel, R.: On the shape of a set of points in the plane. IEEE Transactions on Information Theory 29(4), 551–558 (1983)
5. Edelsbrunner, H., Mücke, E.P.: Three-dimensional alpha shapes. In: VVS, pp. 75–82 (1992)
6. Galton, A., Duckham, M.: What is the region occupied by a set of points? In: Raubal, M., Miller, H.J., Frank, A.U., Goodchild, M.F. (eds.) GIScience 2006. LNCS, vol. 4197, pp. 81–98. Springer, Heidelberg (2006)
7. Gamper, J., Böhlen, M., Cometti, W., Innerebner, M.: Scalable computation of isochrones in bimodal spatial networks. Technical report, Free University of Bolzano-Bozen (2010)
8. Graham, R.L.: An efficient algorithm for determining the convex hull of a finite planar set. Inf. Process. Lett. 1(4), 132–133 (1972)
9. Jarvis, R.A.: On the identification of the convex hull of a finite set of points in the plane. Inf. Process. Lett. 2(1), 18–21 (1973)
10. Moreira, A.J.C., Santos, M.Y.: Concave hull: A k-nearest neighbours approach for the computation of the region occupied by a set of points. In: GRAPP (GM/R), pp. 61–68 (2007)

11. Mouratidis, K., Yiu, M.L., Papadias, D., Mamoulis, N.: Continuous nearest neighbor monitoring in road networks. In: VLDB, pp. 43–54 (2006)
12. Papadias, D., Zhang, J., Mamoulis, N., Tao, Y.: Query processing in spatial network databases. In: VLDB, pp. 802–813 (2003)
13. Zhang, R., Lin, D., Ramamohanarao, K., Bertino, E.: Continuous intersection joins over moving objects. In: ICDE, pp. 863–872 (2008)

CM-Quality: A Pattern-Based Method and Tool for Conceptual Modeling Evaluation and Improvement

Kashif Mehmood[1,2], Samira Si-Said Cherfi[1], and Isabelle Comyn-Wattiau[1,2]

[1] CEDRIC-CNAM, 292 Rue Saint Martin, F-75141 Paris Cedex 03, France
[2] ESSEC Business School, Avenue Bernard Hirsch B.P. 50105, 95021 Cergy-Pontoise Cedex, France
Kashif.Mehmood@essec.fr, {samira.cherfi,isabelle.wattiau}@cnam.fr

Abstract. Conceptual models serve as the blueprints of information systems and their quality plays a decisive role in the success of the end system. It has been witnessed that majority of the IS change-requests result due to deficient functionalities in the information systems. Therefore, a good analysis and design method should ensure that conceptual models are correct and complete, as they are the communicating mediator between the users and the development team. Our approach evaluates the conceptual models on multiple levels of granularity in addition to providing the corrective actions or transformations for improvement. We propose quality patterns to help the non-expert users in evaluating their models with respect to their quality goal. This paper also demonstrates a software utility *(CM-Quality)* that implements the proposed evaluation approach.

Keywords: Conceptual Model Quality, Quality Patterns, Quality Attributes, Quality Metrics, Quality Evaluation, Quality Improvement.

1 Introduction

Information Systems (IS) require high cost for maintenance activities and therefore software quality is considered as an important issue in IS firms. It has now been widely agreed that the quality of the end-system depends on the quality of the Conceptual Models (CM). These CMs are designed as part of the analysis phase and are the basis for further design and implementation. Thus, if there are errors and deficiencies in the CMs then they are propagated along the development process. These errors are more expensive to fix once the system is developed and deployed. For these reasons, different methodologies propose different methods and guidelines to ensure a certain degree of quality to the produced deliverables. These methodologies can include the identification of different design patterns, such as the GRASP (General Responsibility Assignment Software Patterns) design patterns, as they propose solutions to common problems in a given context. However, these design patterns does not explicitly target the quality but propose an expert solution to a common problem. There does not exist any design pattern that integrates the evaluation or improvement of quality of the end system.

In order to incorporate the notion of quality evaluation or improvement within the design patterns, a new concept (named as quality pattern) has recently emerged. It uses the epistemology of design patterns and includes criteria to guide the evaluation of conceptual models and suggestions to improve them. To date, we have identified sixteen quality patterns.

This article aims to propose a quality evaluation and improvement process for conceptual models based on quality patterns. This paper includes both:

- The definition of "quality pattern" concept similarly to design pattern.
- *CM-Quality*: A software utility implementing the proposed approach.

One main advantage of our adaptive quality approach is that it can answer different goal specific quality needs. It is enriched with corrective actions provided to the designer, leading to a guided modeling process. The rest of the paper is organized as follows. Section 2 is a brief state-of-the-art. Section 3 describes our quality model. A software prototype *"CM-Quality"* implementing the proposed approach is described in Section 4. Section 5 concludes and lists the future research directions.

2 Literature Review

Conceptual Models (CM) are the abstraction of the universe of discourse under consideration [1]. They are designed as part of the analysis phase and serve as a communicating mediator between the users and the development team. They provide abstract descriptions and hide the implementation details. Although a CM may be consistent with the universe of discourse, it might not necessarily be correct. This suggests that there is a strong urge for a quality-oriented approach that can help in ensuring the consistency, completeness and correctness of the conceptual models.

Research in software quality is rather mature and has produced several standards such as ISO 9126 and ISO 25030:2007[2, 3] whereas, in the domain of CM, research on quality evaluation is rather young. The first structured approach dates back to the contribution of [4]. They were the pioneers in proposing quality criteria relevant to CM evaluation. In [5], the quality of models is evaluated along the three dimensions: syntax, semantics and pragmatics. In [6, 7], we proposed a more comprehensive model for CM quality evaluation and improvement. However, the selection of the proposed quality criteria (attributes, metrics etc.) can be tricky for a novice user. There is a lack of methodologies putting together the evaluation of quality and its improvement through a real guidance process. [8] argues for employing quality patterns to guide the evaluation process. The idea of patterns was originally proposed by [9] as an architectural concept and was later applied to the computer science programming domain by [10]. However, the concept of patterns proved significantly influential after the advent of Design patterns as they contributed towards the improvement in the software reusability and maintainability as demonstrated by [11]. The concept of quality patterns was first proposed by [12] and it targets the software engineers. Our objective is to propose an analogous approach dedicated to conceptual modeling.

The concept of quality pattern was proposed in [8]. This paper presents a more comprehensive quality approach. It proposes a quality pattern driven evaluation and improvement process for conceptual models. Moreover, an evolutionary software prototype (*CM-Quality*) implementing the overall quality approach is also presented.

3 Proposed Approach and its Expected Contributions

The scope of this paper includes the creation of certain artifacts (quality approach, software utility etc.). Therefore we employed design science as the principal research methodology. Much of the Information systems (IS) research can be classified into either behavioral science or design science research. The behavioral science research seeks to develop and verify theories that explain or predict human or organizational behavior whereas design science seeks to extend the human and organizational capabilities by creating new and innovative artifacts [16]. Similarly designing, improvements and maintenance tasks are largely considered as design science activities. Authors in [16] have defined IT artifacts to be:

i. Constructs (Vocabulary and Symbols),
ii. Models (Abstractions and Representations),
iii. Methods (Algorithms and Practices), and
iv. Instantiations (Implemented and Prototype systems).

The authors in [17] have argued that research aimed at developing IT systems or improving IT practice has been more successful and important than traditional scientific attempts to understand it. Moreover, design science attempts to create technology-oriented artifacts that serve human purposes and its products are assessed against criteria of value or utility such as “Does it work?” or “Is it an improvement?” etc.

The lack of largely used and validated quality frameworks was noticed in [13]. Thus we considered synthesizing (existing concepts proposed by researchers) and adding the new concepts to formulate a comprehensive quality approach for conceptual modeling. This approach encompasses both quality evaluation and improvement in a guided way. The main contributions include: (i), the identification of a set of quality attributes, relevant to both researchers and practitioners, (ii) the definition of “quality pattern” concept similar to design pattern. Sixteen quality patterns, based on validated quality attributes, are already identified; (iii) CM-Quality: A research prototype implementing the proposed approach.

3.1 Identification and Validation of Quality Attributes

Quality attributes can be defined as the group of properties observable over the product lifecycle [18] or the group of properties of the service delivered by the system to its users. Within system engineering domain, quality attributes can also be regarded as the non-functional requirements for evaluating the performance of the system. These attributes are also referred to as “ility” due to the common suffix of many of the quality attributes such as “compatibility, extensibility, modifiability, etc.” [19] defines "ility" as a characteristic or quality of a system that applies across a set of functional or system requirements.

There exist numerous definitions of quality attributes specific to different domains and applications. Within the domain of conceptual modeling quality, researchers have classified their evaluation criteria into dimensions, attributes, characteristics, factors, metrics, etc. There is a clear distinction between metrics and other classification categories due to the widely accepted format of metrics. However, there exists a huge confusion among the definitions of dimensions, attributes, characteristics, factors, etc. Some researchers have defined a concept as a dimension whereas others have used the same definition and called this concept an attribute. For example [20], [21], [22] considered the quality criteria such as correctness, completeness, simplicity and understandability as dimensions whereas the authors in [1] have considered the same criteria as quality attribute.

Similarly, ISO-9126 standard classified quality criteria into characteristics (such as maintainability) and sub-characteristics (such as changeability, testability, customizability). However, these quality characteristics are variously called quality dimensions, factors, principles, criteria, categories, goals etc. Curiously, none of the proposals refer to the ISO terminology [13].

Another main problem in the area of CM quality is the presence of independent quality models not drawing conclusion from similar work. This has resulted in the existence of multiple definitions for the same concept and different names for semantically same concepts. The authors in [23] have identified different definitions of the same quality concepts e.g. they have identified nine different definitions for the quality attribute “completeness”. Similarly, there exist numerous definitions for the same quality concept and identical names for some semantically different metrics [24].

In order to resolve the above mentioned issues, we performed the following:

- Different aspects of conceptual modeling quality are identified and classified into multiple attributes thorough a comprehensive literature review. Each selected attribute is generic and valid for all types of conceptual models.
- The concept of quality attributes in our approach unifies the existing concepts such as dimensions, attributes, characteristics or sub-characteristics, criteria, factors etc.
- Multiple definitions of similar concepts (such as completeness, as mentioned above) are catered by formulating different metrics.
- A selected set of quality attributes was validated using a web-based survey. Respondents were asked to provide feedback over the efficacy of the selected attributes. The description about the quality attributes and the survey results can be consulted from [6, 7].

3.2 Quality Pattern and Quality Oriented Development Process

Quality evaluation is a difficult activity requiring a high level of expertise. Moreover, the proliferation of quality concepts in literature and the absence of agreement about their definition and usage increase this difficulty. After the first step that led to quality attributes selection and validation, we propose to capitalize knowledge and existing “good practices” in the field of quality evaluation and improvement. Indeed, there is a lack of methodologies putting together the evaluation of conceptual models and their improvement through a guidance process. Most of the existing approaches or proposals fail to provide post evaluation improvement guides [13]. This capitalization relies on the concept of Quality Pattern. A quality pattern as presented in [8] is a groundbreaking concept for quality driven modeling process. Similarly, to design patterns, a quality pattern is a reusable solution to a commonly occurring problem in quality evaluation and improvement. It aims to capitalize the expertise required to guide analysts and designers in solving quality problems. Our quality pattern driven approach provides a four-step quality problem approach:

1- Helps in quality goals elicitation,
2- Matches the goal with suitable quality attributes and metrics,
3- Helps in the evaluation of the quality attributes, and
4- Provides recommendations on how to improve quality.

The following sections define with more detail the concept of quality pattern and the quality pattern driven modeling process.

3.2.1 The Proposed Quality Meta-model

Our quality meta-model follows a Goal Question Metric (GQM) [14] approach. It is based on the notion of quality patterns and manages the model quality with respect to user's needs. The meta-model in Figure-1 is generic and simple. A "quality goal" expresses a need to improve the quality of a CM. A quality goal could be related to several quality attributes.

For example, the quality goal "make my CM more extendible" is related to "modularity" and "complexity" quality attributes. Quality attributes are contained in quality patterns that guide their measurement and improvement. Quality attributes are quantifiable through quality metrics. Based on the results of the quality metrics, corresponding predefined transformations and/or appropriate design patterns are proposed for improvement.

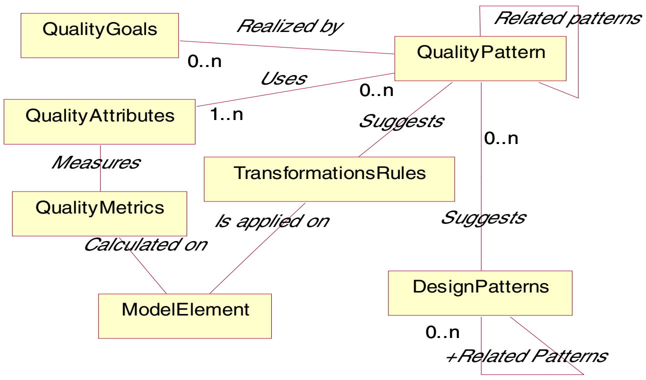


Fig. 1. Proposed Quality Meta-Model

One strength of our proposal lies in the relationship between the quality patterns (as a mean of quality evaluation) and design patterns (as a mean of model transformation). In addition to the search for suitable design patterns whose application is under the responsibility of the designer, our approach provides him/her with a set of transformation rules as a more helpful way for quality improvement.

3.2.2 An Instantiation of the Quality Pattern Meta-model

Currently we have identified sixteen quality patterns based on the above mentioned meta-model. Each quality pattern respects the following outline that has become fairly standard within the software community to structure patterns.

Name: a significant name summarizing the pattern objective.
Context: characterization of the situation in which the pattern applies.
Problem: description of the problem to solve or the challenge to be addressed.
Solution: the recommendation to solve the problem.
Keywords: a list of keywords related to the pattern content
Related patterns: patterns that are closely related to the one described.

Table-1 sketches an example of a quality pattern dedicated to the evaluation and improvement of the simplicity of a conceptual model.

Table 1. Quality Pattern for Model Simplicity

Pattern Name	**Model Simplicity**
Context	There is a need to maintain model simplicity
Problem	Complex models are difficult to understand, implement and maintain. The complexity could be difficult to manage as it could be related to several sources (domain, structure, modeling notation etc.).
Solution	**Design patterns**: High cohesion GRASP pattern, indirection GRASP pattern and polymorphism GRASP pattern. **Transformation rules**: divide a model, merge classes/entities, use factorization mechanism etc.
Keywords	Complexity, Simplicity, Structural Complexity, Size
Related patterns	Model Modifiability; Model Reusability

3.2.3 Quality-Pattern Driven Evaluation Process

Our proposed quality aware methodology aims at helping the achievement of a quality goal formulated by an IS designer. The process starts with the formulation of a quality goal (by the IS designer). The approach helps in the achievement of this goal by identifying and proposing a set of applicable quality patterns. The interpretation of a quality pattern proposes either a set of transformation rules or a set of suitable design patterns leading to the improvement of the CM according to the formulated quality goal.

Our proposed quality aware methodology does not require a long learning process. As depicted in the meta-model, quality patterns are formulated by using the existing quality attributes in the knowledgebase. Quality patterns are created to help the user (beginner or expert) to achieve his/her quality goal efficiently.

The quality-pattern based evaluation process starts with the formulation of a specific quality goal. For example, a user is interested in evaluating a conceptual model with respect to the ease with which it could be changed. This goal is a vague statement. Thus the approach proposes to define it more precisely using questions. For example, for ease of change, we could ask to make clear the kind of change (preventive, perfective or corrective). This refinement of the goal helps in matching it with predefined quality patterns stored in the knowledgebase. . These quality patterns have been identified by extracting and integrating quality evaluation practices from the literature. A first informal validation has been conducted by several experts. We are currently working on a larger validation experiment with students and practitioners in order to evaluate the acceptance of these patterns by novice and expert analysts and designers.

To enhance the flexibility of the process, the approach offers the possibility to select the quality pattern to be used or even the quality attributes to be evaluated. This, however, requires both expertise in quality evaluation and a degree of familiarity with the quality pattern concept. After a quality pattern is matched to the quality goal, the quality pattern drives the evaluation of quality according to the related quality attributes and metrics. The analyst and designer are requested during the process to make one or several choices among those proposed.

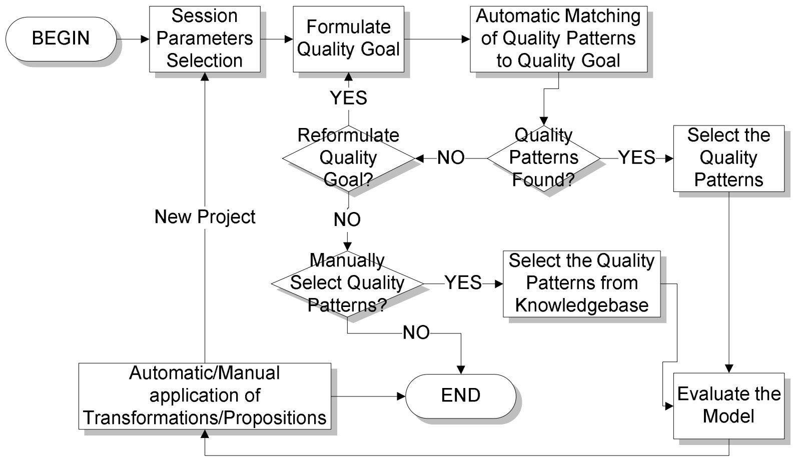

BEGIN
Session Parameters Selection
Formulate Quality Goal
Automatic Matching of Quality Patterns to Quality Goal
YES
Reformulate Quality Goal?
NO
Quality Patterns Found?
YES
Select the Quality Patterns
New Project
NO
Manually Select Quality Patterns?
YES
Select the Quality Patterns from Knowledgebase
NO
Automatic/Manual application of Transformations/Propositions
END
Evaluate the Model

Fig. 2. Quality Pattern Driven Evaluation Process

The above mentioned methodology to identify and link quality patterns with user specific quality goal is summarized in Figure-2.

4 CM-Quality: An Automated Environment Implementing the Proposed Approach

We have designed and developed a prototype, ***"CM-Quality",*** which implements our quality approach. This implementation has two core objectives. It first helps in demonstrating the feasibility of the approach. The second objective is related to the validation of the approach as we plan to make the prototype available to students, researchers, and practitioners to collect their feedbacks.

CM-Quality works along with an independent software utility, *Qualis*, which we developed in a previous research. The two modules of the *CM-Quality* use the services related to metrics definition and calculation offered by the modules of *Qualis* for the evaluation process.

Finally, *CM-Quality* has an import functionality based on XML allowing the evaluation of quality of IS specifications generated by existing commercial and open source CASE tools (Rational Rose, Objecteering, StarUML etc).

4.1 General Architecture

The general architecture includes two main modules to be used by two roles: Quality expert and IS analyst. The quality expert introduces and maintains the quality concepts defined by the quality driven methodology (patterns, attributes, metrics etc.). The IS analyst uses these concepts for IS quality evaluation. The two modules refer to a common knowledgebase (Figure 3).

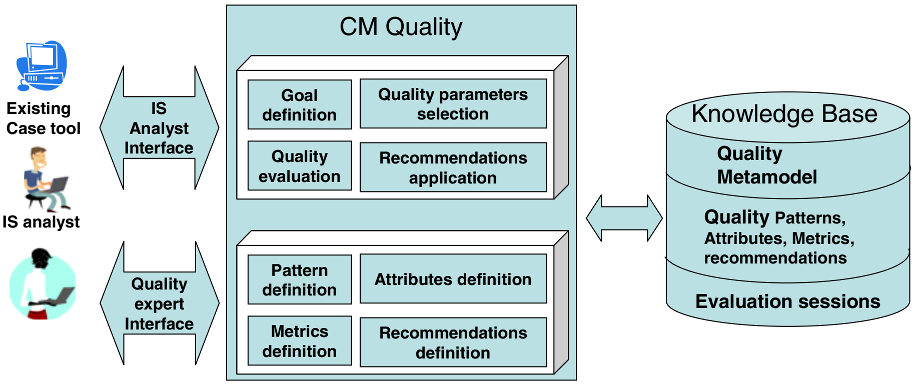


Fig. 3. General architecture of the solution

4.1.1 The Quality Definition Module

The quality expert is responsible for defining the core of the methodology. For this usage, *CM-Quality* proposes four utilities allowing respectively quality pattern definition; quality attributes definition, metrics definition and improvement recommendation definition.

The *Pattern definition* tool guides the definition of quality patterns by proposing predefined quality concepts extracted from the knowledgebase. The quality experts are provided with editors helping the patterns definition according to the pattern definition structure defined in Section 3. The *quality definition* tool provides a rich set of predefined quality attributes with their definition and reference to the literature. It also offers the possibility to add new attributes definition in a guided and structured way. The *metric definition* tool is a complete and complex set of utilities providing both a language and a set of editors for metrics definition. It also provides the expert with a set of predefined quality metrics that could be browsed and modified. Finally, the *recommendation definition* tool enables the definition of corrective suggestions according to a given quality threshold. These recommendations correspond to best practices extracted from literature and validated by IS quality experts.

4.1.2 The Quality Evaluation Module

This module provides an IS analyst with a set of utilities for quality evaluation and improvement. It implements the quality driven process presented in Section 3.

The quality *parameters selection* tool initializes the quality evaluation session. This initialization includes information about the modeling notation used (ER, UML etc.), the specification to be evaluated (class diagram, ER diagram etc.), the specification purpose (implementation, usage, specification) etc. The *goal definition* tool enables quality goal expression. The IS analyst simply describes the desired quality goal in a natural language style. The *quality evaluation* tool includes two main functionalities, namely goal-pattern matching and quality pattern interpretation. The first one is based on text matching techniques performed on the goal expression and quality patterns components such as key words, context description etc. The *quality pattern interpretation* supports the selection of appropriate quality attributes and quality metrics. The evaluation of the quality metrics on the IS specification produces a set of quality values for the selected quality attributes. Finally, based on the obtained quality values, the *recommendation application* tool proposes quality improvement advices. In the future version, this module will implement the transformation rules associated to the recommendations.

4.1.3 The Knowledgebase Structure

The knowledgebase has three abstraction levels. The highest level contains the quality meta-model implementation. The intermediate level is dedicated to quality concepts defined by the quality expert. It contains the quality attributes, metrics and recommendations created throughout Qualis and CM-Quality. Finally, the lowest level stores the results of the evaluation sessions.

4.2 Quality Definition in CM-Quality

CM-Quality is a user-friendly tool aiming to support the quality evaluation approach described in Section 3. As depicted in the meta-model, quality patterns are formulated by using the existing quality attributes in the knowledgebase. Quality patterns serve as guidelines helping IS analysts (naïve or expert) to achieve a quality goal.

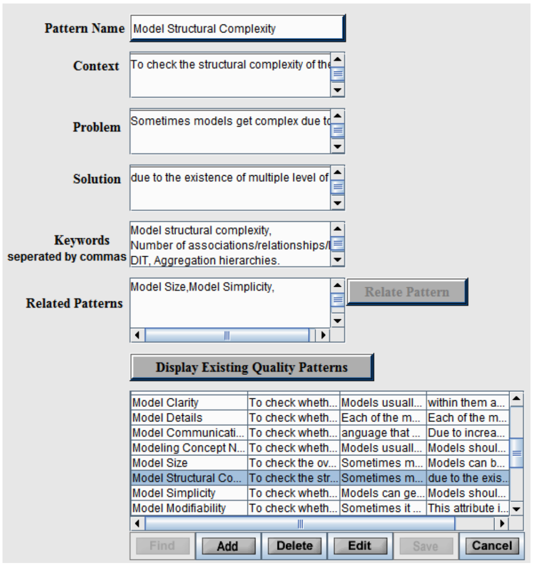
CM-Quality includes two interfaces; the first one, dedicated to quality experts aims to maintain and enrich the knowledge base content. The second, dedicated to quality evaluation and improvement by IS analysts, attempts to match a quality goal with the quality patterns and/or quality attributes stored in the knowledgebase.

4.2.1 Quality Pattern Definition

Defining a new quality pattern requires answering the following three questions:

i. What is the context of this quality pattern or when can this pattern be used?
ii. What is the problem that this pattern can solve?
iii. How can this pattern solve the problem?

Once the quality expert answered the three questions, he/she can use *Quality Pattern interface* to add a new quality pattern to the knowledgebase. Similarly, he/she can use the same interface to edit or delete the quality patterns from the knowledgebase. Figure-4 depicts the *Quality Pattern interface.*



Pattern Name: Model Structural Complexity

Context: To check the structural complexity of the

Problem: Sometimes models get complex due to

Solution: due to the existence of multiple level of

Keywords seperated by commas: Model structural complexity, Number of associations/relationships/I DIT, Aggregation hierarchies.

Related Patterns: Model Size,Model Simplicity,

Model Clarity	To check wheth...	Models usuall...	within them a...
Model Details	To check wheth...	Each of the m...	Each of the m...
Model Communicati...	To check wheth...	anguage that ...	Due to increa...
Modeling Concept N...	To check wheth...	Models usuall...	Models shoul...
Model Size	To check the ov...	Sometimes m...	Models can b...
Model Structural Co...	To check the str...	Sometimes m...	due to the exis...
Model Simplicity	To check wheth...	Models can ge...	Models shoul...
Model Modifiability	To check wheth...	Sometimes it ...	This attribute i...

Fig. 4. CM-Quality : Managing quality patterns

4.2.2 Quality Attributes and Metrics Definition

Analysts can add/edit/delete quality attribute from the knowledgebase using the *Quality Attribute interface*. However, adding a new quality attribute requires its association to the appropriate quality metrics for its quantification. Quality metrics are an important part of the evaluation process. They can be added/edited/deleted from the knowledgebase.

An important strength of our approach is the fact that we have developed in a previous work a prototype (*Qualis*) for metrics definition and evaluation. The metrics are not hard coded providing more flexibility in their definition. This also allows a simple enrichment of metrics. Here is an example of metric definition calculating the number of classes in a class diagram:

```
<metric name="NB_Classes_Metric" domain="model" >
<description>The number of classes belonging to a
model.</description>
<projection  globalrelation="true"  target="class"
condition="id!=''" />
</metric>
```

Finally, the quality expert may define a set transformation rules. Transformation rules are guidelines on how to transform the initial model in order to improve its quality. These guidelines are inspired from best practices in conceptual modeling.

4.3 Quality Evaluation in CM-Quality

The scenario presented in this section illustrates how the *CM-Quality* can be used to evaluate and improve the models. This is a complete scenario that demonstrates the complete flow of the *CM-Quality* application and how it interfaces with *Qualis* for metric calculation and feedback generation.

This scenario doesn't include the housekeeping of knowledgebase i.e. it doesn't discuss about the insertion, modification or deletion of quality patterns, quality attributes, metrics etc to the knowledgebase. It uses the existing contents of knowledgebase for evaluation and propositions. However, *CM-Quality* contains numerous screens for manipulating the knowledgebase including the interfaces to manage quality patterns and quality attributes as shown above.

4.3.1 Goal Expression and Resolution

In this example, the IS analyst wants to improve the quality of his/her conceptual model with respect to complexity. The formulation of quality goals is based on natural language. The CM-Quality tool integrates a searching engine to map the words of the expressed goal and terms associated to patterns quality in several fields such as context, keys words etc. Let's suppose that the user proposed "how complex is my model" as a quality goal.

Matching a goal to quality patterns

The search engine associated to *CM-Quality* uses information contained in patterns and their associated metric to match a quality goal. This matching process could take a lot of time. However, in order to accelerate the searching, the user may select the target fields for searching by restricting the explored fields (one or several among name, context, problem, keywords, quality metrics etc.)

In addition to the automatic detection of quality patterns, the user can select patterns by browsing the content of the knowledge base. He/she could also exclude quality patterns from the resulting ones. This manual selection necessitates expertise in quality evaluation. Figure-5 depicts the automatic searching of quality patterns relevant to the user defined goal.

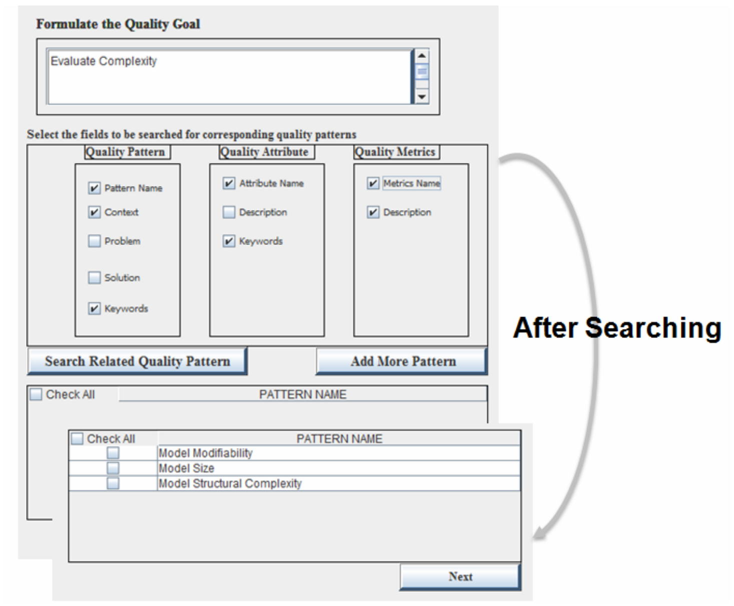
The next step is dedicated to the selection of quality attributes.

Quality Attributes & Metrics Selection

The structure of a quality pattern contains a set of quality attributes. However, the evaluation of all the quality attributes is not mandatory. The IS analyst can change the selection and decide the quality attributes to be used from each quality pattern for his/her goal specific evaluation project. Similarly, the IS analysts can also modify the selection of metrics for their evaluation project. For example, in Figure-6, *"Model Structural Complexity"* is selected as the selected quality pattern and *"Structural Complexity"* is selected as the quality attribute. The table displays all the metrics associated with the selected quality attribute. Among the whole list, only seven metrics are selected for evaluation.

Goal Evaluation

The goal evaluation consists in evaluating the selected metrics on the model defined by the IS analyst. In our example, the model was a UML class diagram. The results could be visualized for a specific model element or globally for the entire model. Due to space constraints we may not provide the reader with the result screen.


Formulate the Quality Goal

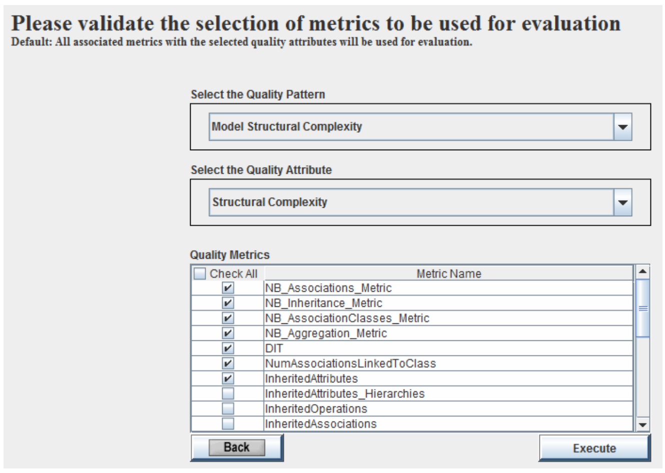
Evaluate Complexity

Select the fields to be searched for corresponding quality patterns

Quality Pattern	Quality Attribute	Quality Metrics
[✓] Pattern Name	[✓] Attribute Name	[✓] Metrics Name
[✓] Context	[ ] Description	[✓] Description
[ ] Problem	[✓] Keywords	
[ ] Solution		
[✓] Keywords		

Check All	PATTERN NAME
[ ]	Model Modifiability
[ ]	Model Size
[ ]	Model Structural Complexity

Fig. 5. Goal Creation: Automatic searching of quality patterns


Please validate the selection of metrics to be used for evaluation

Default: All associated metrics with the selected quality attributes will be used for evaluation.

Select the Quality Pattern: Model Structural Complexity

Select the Quality Attribute: Structural Complexity

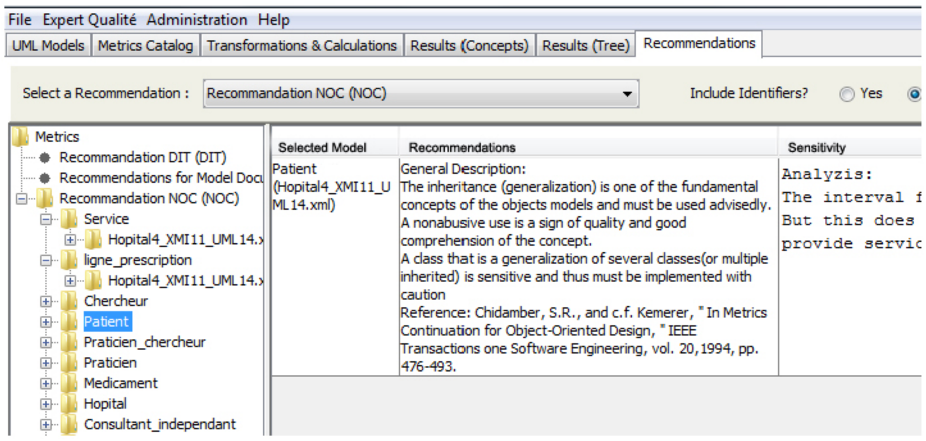
Quality Metrics

Check All	Metric Name
✓	NB_Associations_Metric
✓	NB_Inheritance_Metric
✓	NB_AssociationClasses_Metric
✓	NB_Aggregation_Metric
✓	DIT
✓	NumAssociationsLinkedToClass
✓	InheritedAttributes
	InheritedAttributes_Hierarchies
	InheritedOperations
	InheritedAssociations

Fig. 6. Goal Creation: Validating the metrics selection for evaluation

Post evaluation feedback on the model

As mentioned above, one main quality of our approach lies in the post evaluation feedback in the form of transformations rules or corrective actions. Our knowledge-base contains different propositions or transformations that are available to the user once the model is evaluated. These propositions depend on the metrics values. For example, Figure-7 displays the recommendation explaining in a textual form the



Select a Recommendation : Recommandation NOC (NOC)

Include Identifiers? ○ Yes ●

- Metrics
  - Recommandation DIT (DIT)
  - Recommendations for Model Docu
  - Recommandation NOC (NOC)
    - Service
      - Hopital4_XMI11_UML14.x
    - ligne_prescription
      - Hopital4_XMI11_UML14.x
    - Chercheur
    - Patient
    - Praticien_chercheur
    - Praticien
    - Medicament
    - Hopital
    - Consultant_independant

Selected Model	Recommendations	Sensitivity
Patient (Hopital4_XMI11_UML14.xml)	General Description: The inheritance (generalization) is one of the fundamental concepts of the objects models and must be used advisedly. A nonabusive use is a sign of quality and good comprehension of the concept. A class that is a generalization of several classes(or multiple inherited) is sensitive and thus must be implemented with caution Reference: Chidamber, S.R., and c.f. Kemerer, " In Metrics Continuation for Object-Oriented Design, " IEEE Transactions one Software Engineering, vol. 20, 1994, pp. 476-493.	Analyzis: The interval f But this does provide servic

Fig. 7. Post evaluation recommendations

relationship between the complexity of a model and the number of contained structural elements. In the current version of the tool, the recommendations are not applied automatically on the model under evaluation.

We have applied the approach on a Human Resource (HR) management case study. The specification was a UML class diagram with 46 classes, 174 attributes, 120 operations, 50 associations and 14 inheritance links. The specification has been judged difficult to understand by students. Our quality approach led to decrease complexity by essentially splitting the model into two sub models, namely “Payroll” and “Personnel data” and factorizing associations and operations. Two quality patterns (complexity and modularity) were selected. The validation of the approach using the HR case study is under realization. For lack of space, the example could not be included in this paper.

5 Conclusion and Implications for Further Research

This paper proposes a comprehensive quality-pattern driven evaluation and improvement process for conceptual models. The quality approach is based on a generic and flexible quality meta-model that remains valid for different types of conceptual models (ER models, UML diagrams etc.). The meta-model is simple and could be easily instantiated. The instantiation process produces a tree structure that could be used to incrementally guide an IS designer or a quality engineer in achieving a quality goal. This user specified quality goal is refined through quality patterns or quality attributes and is measured by quality metrics. The last level of the quality aware process suggests the possible transformations that propose actions leading to the quality improvement according to the desired quality goal. These quality patterns, representing both researcher’s and practitioner’s quality practices, are capitalized in an evolutionary knowledgebase.

Our approach is based on existing conceptual modeling literature as its theoretical foundation to formulate a multi faceted quality knowledgebase. Different concepts, from the previously existing quality frameworks or literature, are extracted and filtered to construct the proposed knowledgebase. More details on the contents of the knowledgebase can be consulted in [6, 7].

CM-Quality, a software utility was developed to implement the proposed a semi-automatic quality approach. Users can state their quality goals using CM-Quality and the utility will guide them in formulating their quality project by proposing the corresponding quality patterns that are in line with the stated quality goal. After the verification and selection of the quality criteria, the utility evaluates the model on the selected criteria and suggests an improvement strategy (in the form of propositions or transformations) for conceptual models.

Future directions of this work include:

- The extension and enrichment of the current quality model;
- A more comprehensive validation of the approach employing a suitable research methodology.

References

1. Cherfi, S.S., Akoka, J., Comyn-Wattiau, I.: Measuring UML Conceptual Modeling Quality-Method and Implementation. In: Pucheral, P. (ed.) Proceedings of the BDA Conference, Collection INT, France (2002)
2. ISO/IEC 9126, Software Engineering - Product quality - Part 1: Quality model (2001)
3. ISO/IEC 25030:2007, Software Engineering - Software Product Quality Requirements and Evaluation (SQuaRE) - Quality Requirements, Int'l Organization for Standardization (2007)
4. Batini, C., Ceri, S., Navathe, C.: Conceptual Database Design: An Entity-Relationship approach, p. 496. Benjamin/Cummings Publishing Company Inc. (1992)
5. Lindland, O.I., Sindre, G., Sølvberg, A.: Understanding Quality in Conceptual Modeling. IEEE Software 11(2), 42–49 (1994)
6. Mehmood, K., Cherfi, S.S., Comyn-Wattiau, I.: Data Quality Through Model Quality: A Quality Model for Measuring and Improving the Understandability of Conceptual Models. In: DQS 2009, International Workshop on Data Quality and Security in Conjunction with (CIKM 2009), Hong Kong, November 2-6 (2009)
7. Mehmood, K., Cherfi, S.S., Comyn-Wattiau, I.: Data Quality through Conceptual Model Quality - Reconciling Researchers and Practitioners through a Customizable Quality Model. In: ICIQ (Intl. Conference on Information Quality), Germany, November 7-8 (2009)
8. Cherfi, S.S., Akoka, J., Comyn-Wattiau, I.: Quality Patterns for Conceptual Modeling. In: Li, Q., Spaccapietra, S., Yu, E., Olivé, A. (eds.) ER 2008. LNCS, vol. 5231, pp. 142–153. Springer, Heidelberg (2008)
9. Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fiksdahl-King, I., Angel, S.: A Pattern Language: Towns, Buildings, Construction. Oxford University Press, USA (1977)
10. Beck, K., Cunningham, W.: Using Pattern Language for Object-Oriented Programs. In: OOPSLA 1987 Workshop on the Specification and Design for Object-Oriented Programming (1987)
11. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, Reading (1995)

12. Houdek, F., Kempter, H.: Quality Patterns – An approach to packaging software engineering experience. ACM SIGSOFT Software Engineering Notes 22(3) (1997)
13. Moody, D.L.: Theoretical and Practical Issues in Evaluating the Quality of Conceptual Models. Data & Knowledge Engineering 55, 243–276 (2005)
14. Basili, V.R., Gianluigi, C., Rombach, H.D.: The Goal Question Metric Approach. In: Encyclopedia of Software Engineering. Wiley, Chichester (1994)
15. Larman, C.: Applying UML and Patterns: an Introduction to Object-Oriented Analysis and Design and the Unified Process, 2nd edn. Prentice Hall, Englewood Cliffs (2001)
16. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design Science in Information Systems Research. MIS Quarterly 28(1), 75–105 (2004)
17. March, S.T., Smith, G.F.: Design and Natural Science Research on Information Technology. Decision Support Systems 15, 251–266 (1995)
18. Preiss, O., Wegmann, A., Wong, J.: On Quality Attribute Based Software Engineering. In: Proceedings of the 27th EUROMICRO Conference (2001)
19. Manola, F.: Providing Systemic Properties (Ilities) and Quality of Service in Component-Based Systems. Object Services and Consulting, Inc., Technical Report (1999)
20. Moody, D.L., Shanks, G.G.: Improving the quality of data models: empirical validation of a quality management Framework. Information Systems 28(6), 619–650 (2003)
21. Moody, D.L., Shanks, G.G., Darke, P.: Strategies for improving the quality of entity relationship models. In: Information Resource Management Association (IRMA) Conference, Idea Group Publishing, USA (2000)
22. Moody, D.L., Shanks, G.G.: What makes a good data model? A framework for evaluating and improving the quality of entity relationship models. Australian Computer Journal (1998)
23. Nelson, R.R., Todd, P.A.: Antecedents of Information and System Quality: An Empirical Examination within the Context of Data Warehousing. Journal of Management Information Systems 21(4), 199–235 (2005)
24. Purao, S., Vaishnavi, V.: Product Metrics for Object-Oriented Systems. ACM Computing Surveys 35(2), 191–221 (2003)

A Performance Analysis of Semantic Caching for Distributed Semi-structured Query Processing[⋆]

Boris Novikov, Alice Pigul, and Anna Yarygina

Saint-Petersburg University
borisnov@acm.org, alica_pigul@bk.ru, anya_safonova@mail.ru

Abstract. Caching is important for any system attempting to achieve high performance. The semantic caching is an approach trying to benefit from certain semantical knowledge of the data to be processed.

The expectation is that semantical information might help to reduce the number of cache misses and in certain cases even avoid queries to the primary data. However, the major obstacle for wide application of semantic caching is the query containment problem which is computationally hard.

In this paper we introduce an approximate conservative algorithm for semantic caching of semistructured queries and analyze its applicability for distributed query processing. Based on this analysis, we outline few scenarios where semantic caching can be benefitial for query processing in a distributed system of heterogeneous semi-structured information resources.

1 Introduction

The need to uniformly access heterogeneous information resources triggered a huge amount of research in the area of information integration performed in last decades. Several industrial strength integration software tools are currently available. However, the query evaluation performance is still far from practically acceptable for any complex queries and is often unpredictable. Common techniques for performance improvement, such as indexing, query optimization, and caching, should be revised and re-evaluated in the distributed heterogeneous environment.

The focus of this research is on advanced caching techniques for distributed heterogeneous systems, call semantic caching. In contrast with usual caching techniques, in addition to the usage patterns, semantic caching takes into account relationships and inter dependencies between data items to improve cache effectiveness and enable local query processing of cached data instead of distributed query processing.

[⋆] This research is supported by HP Labs Innovation Award program (project: HPL-800481-20080624052858282) and Russian Foundation for Basic Research, grant 10-07-00156.

The goal of this research is to evaluate the potential impact of different approaches to semantic caching and obtain quantitative estimations of their suitability.

The idea of caching is to reduce the number of repeated accesses to the data items located in relatively slow large storage. Instead, copies of data items are temporarily copied into fast (and relatively small) memory.

In almost all implementations cache is completely transparent for the application or system that uses the cached data. That is, the presence of cache affects performance, but not the functionality of the application, and, moreover, the application cannot explicitly control the behavior of the cache.

In this research, the data items copied from the primary memory into cache are called cache units.

In a common caching the cache units are pages, blocks, or other low-level data structures defined for the physical storage. However, this is not applicable for distributed environment because low level units are not related to the units transferred over a network, and high level objects, such as web pages, are cached instead. A potential disadvantage is that the cache unit is considered atomic and can be used only for exactly the same request.

The caching is called semantic caching if the cache units are meaningful items or sets in the high-level application data model. Thus, parts of cache unit may be used to process queries other than original one, or data from several cache units may be used together to evaluate new query.

The expectation is that the above mentioned should improve reuse of the cached data. On the other hand, the extraction of cached data becomes more complex and may result in certain performance penalties. Hence, our goal is to find conditions under which the semantic caching is benefitial.

A significant amount of work have been done during the last two decades.

It turns out that the major restricting factor for effective use of semantic caching is the query containment problem which is proven to be decidable only for limited class of queries and has polynomial complexity for much smaller class of queries.

To make the semantic caching efficient, one has to restrict the class of cached queries and use approximate algorithms for query containment problem. If an approximate algorithm can produce false positive answers, the use of cache may result in data loss in the query responses.

An algorithm for query containment problem is called conservative if it never returns false positives.

We consider only conservative algorithms in this research.

The rest of the paper is organized as follows.

We provide an overview of this work in the next section. We then outline the detailed goals of our study and describe experimental environment, followed by analysis of experimental results. Based on this analysis, we outline few use cases where semantic caching is beneficial. More details on experiments are included as an appendix.

2 Related Work

The notion of semantic caching was elaborated in numerous works including [11,19,14,2,27,5]. Among past research areas related to semantic caching there are query containment and satisfiability problems [16], view materialization [21,1], query folding [26] and semantic query optimization [9].

Franklin et al. in [11] proposed to use the queries that have been used to obtain the results in the cache as high level descriptors. According to [11] the entries in the semantic cache are organized by semantic regions. Semantic regions are defined dynamically based on answered queries. Each semantic region has a constraint formula (high-level description).

The authors of [19] suggested to reduce the number of semantic regions by merging them whenether possible. The same idea was used in [2]. They presented an approach for systematically creating new semantic region to consider for caching and merging them when possible.

The major obstacle for wide use of semantic caching in relational systems is the query containment problem complexity. It forces to limit the class of predicates to conjunctive predicates with simple range conditions.

Deciding whether a query is answerable or not is similar to finding complete rewritings of a query using views ([21],[26],[20]). The main difference is that rewriting techniques transform a given query based on the views [10], while semantic caching evaluate a given query according to semantic views ([11]).

Linear constraint-based semantic caching, based on the idea of constraint databases, representation method of cache information and the cache examination algorithm were considered in [17]. The design and the evaluation of the ADMS optimizer were described in [22]. Efficiently and effectively solving the satisfiability problem, i. e. whether there exists a contradiction in a formula consisting of conjunctive inequalities was studied in [15].

Semantic caching, a caching architecture for online data services and e-commerce applications , that caches the results of selection queries were considered in [18].

The XPath language to navigate XML documents and extract information from them was used in ([32],[33]), XQuery language was studied in ([23],[31],[29]).

How to decide that the cached information is sufficient to answer a query (that is, the cache contains needed data), and how to incrementally maintain the cached XML document was discussed in [32] . It also shows how represent cached XML data by a set of XPath queries.

The problem of rewriting XPath queries using materialized XPath views,i.e. whether there exists a rewriting of a query using XPath views and the problem of finding minimal rewritings, were considered in [33]. The cached data are organized as modifided incomplete tree in [31]. The authors also describe an algorithm for construction of reminder query to extract data which are not contained in the cache, from the primary data source.

Techniques for caching frequent query patterns are introduced in [29]. The paper presents an algorithm to generate query subpatterns, check query containment, and choose query subpatterns for caching.

The problem of partial query matching was studied in [23]. A semantic cache architecture was presented in ([23],[32],[31],[29]) .

A huge volume of theoretical and implementation work was done in the last decade on semantics representation based on onologies. A significant part of this work is related to RDF and OWL languages, and some of papers address performance issues [12,25,3]. However, we study semantic-driven caching of data, rather than caching of semantics definitions in this research. Also, several authors consider triple-based storage extremely inefficient for bulk processing in any industrial environment [13].

In [24] a scheme for reducing the latency perceived by users by predicting and prefetching files that are likely to be requested soon, while the user is browsing through the currently displayed page was investigated. A content-blind query caching middleware for hosting Web applications, which stores the query results independently and does not merge different query results was considered in [30].

A mechanism for efficient caching of Web queries and the answers received from heterogeneous Web providers was used in ([7],[6]). Caching mechanism for conjunctive Web queries based on signature files was used in [4].

The replacement policy was considered essential for any caching technique.

The effectiveness of a cache replacement policy depends on data distribution and access patterns. It is known that widely used LRU replacement policy is not optimal for query processing. More complicated policies, based on the knowledge of access patterns, are described in [28], [8].

In the context of semantic caching the replacement becomes a hard problem because data associated with different cache units may intersect and hence cache clean-up becomes computationally hard. For this reason, typically semantic cache prototypes just drop the whole cache contents, instead of partial replacement. As result, the semantic cache can only be useful if the ratio of primary memory size to cache size is not too high.

3 Targets, Goals, and Experimental Environment

This section describes the target architecture and how caching engine can be used to improve the performance, the goals of the study and experimental environment.

3.1 The Caching Engine

Our target in this research is cache support for query processing in the distributed heterogeneous context with autonomous resources (mealing higher degree of uncertainty, sometimes absence of known schema or reliable statistics).

The architecture of such environment, may, of course, be very complex. However, we assume that all heterogeneous information resources accessible to the query processing engine are hidden behind wrappers and mediators which provide a semi-structured representation of these resources in the XML format. These resources are still heterogeneous as we do not need any assumptions restricting structure, schema, or other constraints on data. However, we further assume that all queries are represented in XQuery language, both from the client and to wrappers or mediators.

The major architectural unit considered in our research is caching engine.

A caching engine is a process that accepts queries to a certain data source, retrieves the data from this data source and, in addition to returning it to request, stores them in the internal cache to be used for processing of subsequent queries.

The caching engine is not responsible for any kinds of query routing to the data sources, as well as for merging data from different data sources (except for merge of cached data with data coming from primary source). For this reason, the caching engine does not depend on the distributed or heterogeneous nature of the environment. In our experiments, we run a model of caching engine as a stand-alone query processor.

Of course, the content of the data source, query patterns, and statistical properties of query sets depend heavily on the type of data source and on the application.

A minimal unit of data from the primary data source that can be stored and accounted in the cache is called *cache unit.* Each cache unit stored in the cache is identified by a *cache unit descriptor.*

For example, in traditional database server cache a cache unit typically is a page and the cache unit descriptor is the page address on the hard drive. For web caching the cache unit might be a downloaded web page and the URL might be used as a descriptor.

In our model the cache unit is the result of the query evaluation, and the query itself represents the semantics of this result and serves as a cache unit descriptor.

Of course, the query results are not as simple as database pages. The results of different queries may intersect, and merged results of several queries may contain also data need to process other queries.

To decide if a given query can be processed in the cache only or access to the primary data source is needed, the query is compared with cache unit descriptors, to find out if the needed data are available in the cache. The answer must be found based on the query text only, without any evaluation, that is, a query containment problem must be solved.

To avoid computational complexity, we restrict the class of queries accepted by the caching engine to very simple XPath expressions for which it is known that the query containment problem can be solved in polynomial time [32]. This class of queries is, however, too restrictive to be useful in practise. For this reason, we extended the class of accepted queries to more complex expressions and use approximate algorithm fo solve the query containment problem. Specifically we allow range predicates at the last step in the XPath query.

All other queries are passed to the primary data source without any processing in the cache.

3.2 Updates and Replacement

Processing of the updates of data located at the primary data source are critical for effectiveness of any cache. Indeed, the cache can be completely transparent only if it contains exactly same data as the primary source. Consequently, the updates of the primary data store must be propagated to the cache, and, if an application updated the cache, these updates must be propagated to the primary data store.

These needs have motivated a huge amount of research as well as industrial solutions for problems of cache synchronization and cache coherence which relates to multiple cache locations for single primary data store (e.g. in shared memory multiprocessor systems, distributed middle ware sharing a database etc.)

Technically the problem of cache synchronization can be splitted into two almost independent problems:

– Update awareness
– Cache replacement.

That is, the information on the updates of the primary data source should somehow reach the caching engine. If the primary resource is using query-response model, then this information may be obtained only via polling the data source periodically.

This hardly can be considered feasible in our heterogeneous environment because wrapped or mediated resources tend to be slow and this kind of polling would jeopardize any potential benefits of caching.

Alternatively, if the server is based on publish-subscribe model, the update notifications can be received and processed in the caching engine as the updated data are requested from the server when they are actually updated.

As soon as the updates are received, the obsolete data must be removed from the cache. There is no need to download new values as this will happen when these data will be requested by the application next time. In other words, lazy approach might be good here. We do not elaborate this topic futher in this research.

For semantic caching the engine must decide which cache units are affected by the update and hence must be removed. Further, data items that do not belong to the remaining cache units can be deleted from the cache. The latter is exactly same problem as incremental cache replacement. Although this problem is not computationally hard, it might require significant amount of query evaluations. For this reason, most of proposed semantic cache implementations do not do any incremental updates of the cache. Instead, they simply clean all data from the cache when it is overflowing. We use the same strategy for this project.

3.3 The Metrics

The quantitative metrics to be experimentally measured are the following:

Cache misses percentage shows how many of incoming queries are sent to the primary data source.

Saturation For a given rate of cache misses shows the size of cache, which is capable to store all data from the hot set.

Precision Shows the percentage of false negatives when solving query containment problem for cache unit descriptors, that is, shows how imprecise the approximate algorithm is.

All metrics listed above depend on the primary data source size and cache size.

3.4 The Experimental Environment

The environment includes several XML (XQuery) processing engines. These engines are used to (1) as a cache storage, (2) cache descriptor storage, (3) Primary data source, (4) merge query results (if needed).

Each experiment runs as it is controlled as the following pseudo-code:

```
begin
while (get next incoming query returns a query)
loop
-- extract simple expressions from the incoming query stream
Match simple (sub)queries against cache descriptors
Case
when subquery is matched successfully
   submit the (sub)query to the cache database
   collect the output
when NOT matched
  submit the query to primary DB
  collect results
  store results to the cache DB
  store the query as a descriptor
end case
merge results of (sub)queries
end loop
exit
```

We used pre-existing database engines, eXist in our experiments. The reasons to use this extremely simple implementation of XQuery were pre-existing expertise and the assumption that absolute performance is not essential for the model experiments.

3.5 Cache Unit Descriptors

For semi-structured (XML) data sources we consider two representations of descriptors and matching algorithms in this research:

– String-based representation and simple matching
– Tree representation with recursive matching.

Both algorithms are conservative as defined above, however, they differ in precision, computational complexity and hence speed of computations.

In the first approach, we use the string presentation of the query in 'a normal' form to describe the processed query in cache. There are different string-based representations of the same query. As result of normalization we can reduce semanticaly equal and syntactically different queries to the same form. In the second representation of descriptors syntactically different but equivalent queries are mapped to the same tree. Consequently, for both representations the syntactical variations do not affect the behavior of the cache.

The matching algorithm for the string representation compares the query in question with stored cache unit descriptors and, if a cache unit descriptor coincides with a prefix of the query to be processed, then the data can be obtained from the cache, otherwise the query is forwarded to the primary data source.

This algorithm can be implemented efficiently as the string prefix mathcing can be supported with database indexes and range scans. Hence the main advantage of this implementation of descriptors in cache is in high speed of computations in match algorithm.

This algorithm does not attempt to recover the structure of the primary data source and therefore we expect the precision of this algorithm is not high.

The second implementation of cache unit descriptors is based on more complex data structure.

We present each processed query by branch in description tree with XPath steps represented as nodes of the tree, i.e. the description tree defines all queries which can be retrieved from cache without any access to the primary data source.

Essentially this tree represents a descriptive schema of the part of the primary data source retrieved by cached queries.

To check that the current query can be answered from the cache we use the recursive tree descent. If the path in description tree which represents the current query is found then the match algorithm returns true.

The query in considered as its tree representation during the homomorphism construction and there is only one representation for the query. The answer to the question if there is full answer to the query in cache could not always been given because of special steps in the query. For example, if there is answer to the query a/b/c in the cache and the current query is //c, the answer to the containment problem could not be done. This proves that our algorithm is approximate.

This algorithm takes some time to construct, increment the tree of descriptors and to decide the containment problem but it is more precise than the previous one.

Most of the experiments described below were performed with the tree-based implementation of descriptors.

3.6 The Data Sets

Two main opportunities for data sets were considered:

1. Generated data based on xMark benchmark definition
2. XBRL reports

The set of experiments described in this research is based on xMark generated data sets.

We rely on XMark data generator and use a simple XPath queries derived from 20 benchmark queries of xMark.

The queries are restricted to the following types of fragments of XPath:

- $XP^{\{/,//,[]\}}$
- $XP^{\{/,*,[]\}}$ and
- $XP^{\{/,//,*\}}$

Our queries are constructed from steps:

– '//' - descendant axes,
– '[]'- branches
– '*' - wildcards
– '/' - child axes.

Acceptable query consists of any two of first three steps and optionnally fourth.

We allow queries to contain simple predicates (comparisons and range predicates). To avoid NP-hardness we will consider the queries with exactly one predicate at the last step. This extension does not effect the algorithm complexity. The containment problem is reduced by recursive descent to the tree homomorphism ([32]).

We alleviate complexity problems by implementing an approximate cache, such as conservative cache.

4 Experimental Results

The first set of experiments was actually performed on synthetic data (generated from xMark definition). The original plan was to run the randomized series of queries on a varying database size from few megabytes to few gigabytes, with range of cache sizes.

For each combination of the primary data source size and cache size a randomized series of queries were run several times, and the average values and dispersion were calculated.

Due to performance limitations, actually the experiments were run only on tiny sizes of the primary data source and also very small cache sizes.

Typical charts drawn from the experimental data are included as an appendix.

The results of experiments are discouraging. First of all, the saturation is reached only when significant portion of the primary source data are fetched into the cache. This, however, is due to completely random generation of the queries with uniform distribution. The recommendation is to use the cache only if the distribution of queried data is not uniform, that is, only if the existence of relatively small hot set can be expected.

Although the absolute performance of the XQuery engine does not affect the measured characteristics, in practice it is prohibitevely poor. Moreover, many experiments were stopped because the capacity of the XQuery engine has been exceeded.

The approximate algorithm for query containment problem, used in this research, can handle even more complex XPath expressions than those suggested theoretically based on complexity of the exact algorithm.

As our algorithm is conservative, the only important measure of quality is its precision, which stays at the acceptable level in our experiments. However, the precision decreases as the complexity of the query grows.

For the range and predicate queries the expected results do not differ significantly from those for relational databases.

For overall performance, the cache may be estimated as follows.

Let

- T be the (average) query processing time on the primary data source,
- t be the (average) query processing time on the cache engine,
- m be the percentage of cache misses (in the range 0-1),
- c be the time for cache overhead (query containment etc.)

Obviously, the expected speed-up may be estimated as $T/(mT+(1-m)t+c)$. It is clearer that high performance gains may be expected only if $T >> t$, $T >> c$ and m is small.

Unfortunately, our experiments show that native XML databases are not designed for high performance on any significant volume of data, especially for complex queries. For this reason, we hardly can expect that T, t, and c will differ sufficiently. Distribution is not the major factor contributing to the value of T.

The poor performance of XQuery engines was also observed on high-end industrial strength database management systems which deliver excellent perfromance on relational data. This suggests that XQuery engines cannot be used as a base for caching engine due to unacceptably high cost of both query matching and data extraction from the cache.

Let us' outline few scenarios when semantic caching of semi structured data sources can still be useful.

4.1 Cascade Architecture

The practice of building new system on top of existing systems inevitably leads to cascade architecture where sub queries are routed to the underlying query processing engins.

The optimization is really hard for this kind of systems. For this reason, it should be expected that certain queries may be submitted several times, especially at the lower levels of the system.

If the query processing engins do not cache the output, the caching engine can eliminate multiple execution of these queries.

4.2 Complex XQuery Views and Functions

In this scenario, the complexity of the primary data source is hidden behind a function (say, XQuery function). Externally this data source will be represented, in extreme case, with single XML node parametrized with attributes:

```
<FunctionName param1="value" param2="value">
...
</FunctionName>
```

An attempt to extract an element with specific attribute values is then interpreted in a mediator or query processor on top of mediator as a function call.

In this scenario, the complexity of data is hidden in the function and hence affects the access time to the primary data source but not the cache.

Of course, this pattern can be used several times at different levels of the XML tree.

4.3 High Performance Database Engine in the Cache

Although the structure of the cached data in the above scenario is very simple, the amount of cached data may be huge as each set of parameter values will be, most likely, cached as a separate cache unit.

Probably it makes sense to use a high-performance database engine to store cache units as it can provide really high performance of the cache. This approach, however, will work if the internal structure of the caching engine will use relational representation of data.

A prototype of the caching engine based on a combination of the second and third scenarios outlined above was implemented for processing of complex financial documents represented in XBRL format. For reasons of computational efficiency, the cache was implemented on top of relational database and very simple conservative matching algorithm was used.

This type of caching engine provides drammatic performance improvements.

5 Conclusions

In this research, we presented an analysis of existing approaches to semantic caching in several environments. The usefulness of any semantic caching system is limited due to complexity of the query containment problem.

Starting from the knowledge gained from previous research, we developed a model of semantic caching engine for distributed heterogeneous environment of semi-structured data sources and experimentally evaluated its performance. We implemented an efficient approximate conservative algorithm for query containment problem.

Although quantitative metrics measured on our model meet, in general, our expectations, the extremely poor performance of native XML databases used inside the caching engine makes the whole approach useful only in very special cases.

We outlined few scenarios where semantic caching of semi-structured XML data can still be useful. Preliminary experiments related to these scenarios show drammatic performance improvements. However, additional experiments are still needed.

References

1. Larson, P.A., Yang, H.Z.: Computing queries from derived relations: Theoretical foundation. Technical report (1987)
2. Ashish, N., Knoblock, C.A., Shahabi, C.: Intelligent caching for information mediators: A kr based approach. In: KRDB, pp. 3.1–3.7 (1998)
3. Bizer, C., Schultz, A.: Benchmarking the performance of storage systems that expose sparql endpoints. In: 4th International Workshop on Scalable Semantic Web Knowledge Base Systems, SSWS 2008 (October 2008)
4. Chidlovskii, B., Borghoff, U.M.: Signature file methods for semantic query caching. In: Nikolaou, C., Stephanidis, C. (eds.) ECDL 1998. LNCS, vol. 1513, pp. 479–498. Springer, Heidelberg (1998)
5. Chidlovskii, B., Borghoff, U.M.: Semantic caching of web queries. The VLDB Journal 9(1), 2–17 (2000)

6. Chidlovskii, M., Roncancio, C.: Semantic cache mechanism for heterogeneous web querying (1999)
7. Chidlovskiiy, B., Roncancioz, C., Schneidery, M.l.: Optimizing web queries through semantic caching (1999)
8. Chou, H.-T., DeWitt, D.J.: An evaluation of buffer management strategies for relational database systems. In: VLDB, pp. 127–141. Morgan Kaufmann, San Francisco (1985)
9. Chu, W.W., Chen, Q., Hwang, A.: Query answering via cooperative data inference. J. Intell. Inf. Syst. 3(1), 57–87 (1994)
10. Cluet, S., Kapitskaia, O., Srivastava, D.: Using ldap directory caches. In: PODS, pp. 273–284. ACM Press, New York (1999)
11. Dar, S., Franklin, M., Jonsson, B., Srivastava, D., Tan, M.: Semantic data caching and replacement. In: VLDB 1996: Proceedings of the 22th International Conference on Very Large Data Bases, pp. 330–341. Morgan Kaufmann Publishers Inc., San Francisco (1996)
12. Erling, O., Mikhailov, I.: Rdf support in the virtuoso dbms. In: CSSW, pp. 59–68 (2007)
13. Faroult, S., Robson, P.: (2006)
14. Godfrey, P., Gryz, J.: Answering queries by semantic caches. In: Bench-Capon, T.J.M., Soda, G., Tjoa, A.M. (eds.) DEXA 1999. LNCS, vol. 1677, pp. 485–498. Springer, Heidelberg (1999)
15. Guo, S., Sun, W., Weiss, M.A.: Solving satisfiability and implication problems in database systems. ACM Trans. Database Syst. 21(2), 270–293 (1996)
16. Guo, S., Sun, W., Weiss, M.A.: Addendum to "on satisfiability, equivalence, and implication problems involving conjunctive queries in database systems". IEEE Trans. Knowl. Data Eng. 10(5), 863 (1998)
17. Ishikawa, Y., Kitagawa, H.: A semantic caching method based on linear constraints. In: Proc. of International Symposium on Database Applications in Non-Traditional Environments, DANTE 1999 (1999)
18. Jonsson, B.T., Arinbjarnar, M., Thorsson, B., Franklin, M.J., Srivastava, D.: Performance and overhead of semantic cache management. ACM Trans. Interet Technol. 6(3), 302–331 (2006)
19. Keller, A.M., Basu, J.: A predicate-based caching scheme for client-server database architectures. The VLDB Journal 5, 35–47 (1996)
20. Levy, A.Y.: Answering queries using views: A survey. Technical Report, VLDB Journal (2000)
21. Levy, A.Y., Mendelzon, A.O., Sagiv, Y.: Answering queries using views (extended abstract). In: PODS 1995: Proceedings of the Fourteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, pp. 95–104. ACM, New York (1995)
22. Melvin, C., Roussopoulos, C.N.: The implementation and performance evaluation of the adms query optimizer: Integrating query result caching and matching (1994)
23. Vaidehi, V., Sumalatha, M.R., Kannan, A.: Xml query processing - semantic cache system (2007)
24. Padmanabhan, V.N., Mogul, J.C.: Using predictive prefetching to improve world wide web latency. Computer Communication Review 26, 22–36 (1996)
25. Pérez, J., Arenas, M., Gutierrez, C.: Semantics and complexity of sparql. ACM Trans. Database Syst. 34(3), 1–45 (2009)
26. Qian, X.: Query folding. In: Proceedings of the 12th International Conference on Data Engineering, pp. 48–55. IEEE Computer Society, Los Alamitos (1996)

27. Ren, K.V.Q., Dunham, M.H.: Semantic caching and query processing. IEEE Transactions on Knowledge and Data Engineering 15, 192–210 (2003)
28. Sacco, G.: Index access with a finite buffer space. In: VLDB14, pp. 301–310 (1988)
29. Shen, H.T., Li, J., Li, M., Ni, J., Wang, W. (eds.): APWeb Workshops 2006. LNCS, vol. 3842. Springer, Heidelberg (2006)
30. Sivasubramanian, S., Pierre, G., van Steen, M., Alonso, G.: GlobeCBC: Content-blind result caching for dynamic web applications. Technical Report IR-CS-022, Vrije Universiteit, Amsterdam, The Netherlands (June 2006)
31. Petropoulos, M., Hristidis, V.: Semantic caching of xml databases (2002)
32. Xu, W.: The framework of an xml semantic caching system. In: WebDB, pp. 127–132 (2005)
33. Xu, W., Ozsoyoglu, Z.M.: Rewriting xpath queries using materialized views. In: VLDB, pp. 121–132 (2005)

Appendix: Measurement Results Summary

This appendix contains a summary of results measurement represented as charts.

The figure 1 shows the behavior of the cache after a cold start or cache clean-up. As expected, the pecentage of cache misses decreases when the number of processed queries grows.

The lines represent behavior of the cache for simple and complex queries. For example, $/site/people/person/name$ is considered simple while $/site/*/open\_auction/*/personref$ and $/site//open\_auction//*/increase$ are considered complex. The number of cache misses for complex queries is less than for simple ones as the data fetched for simple queries is reused for complex ones.

The figure 2 show how many times the cached data was re-used for subsequent queries. This measurement shows how hot the cached data are.

The figure 3 shows the dispersion of experimental data used to produce the charts above, that is, how reliable our measurements are.

The figure 4 shows the precision of our conservative matching algoritm (the complex one, based on tree-representation of the cache descriptors).

The precision is calculated as a ratio of a difference between the total number of processed queries and false negatives to the total number of processed

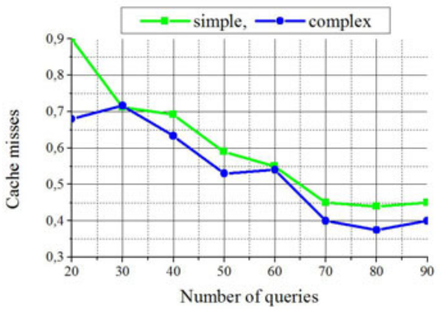


Fig. 1. Cache misses

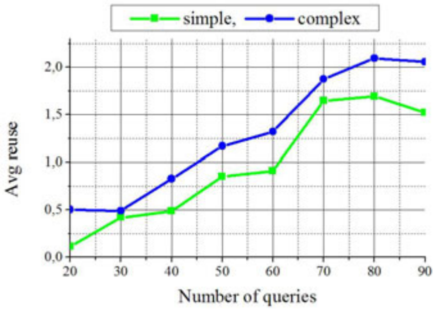


Fig. 2. The reuse of cached data

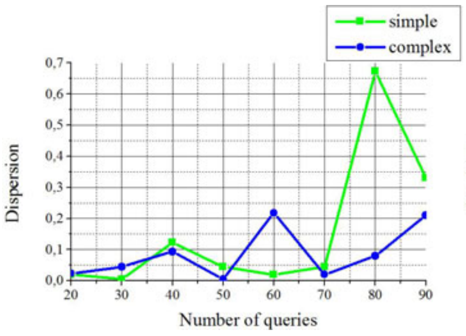


Fig. 3. The dispersion

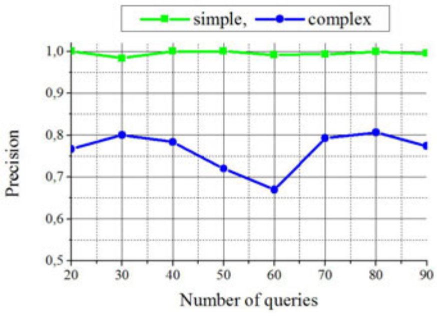


Fig. 4. The precision

queries.To find false negatives, we evaluage the queries against the cache and compare the result with one obtained from the primary data source. Our results show that the precision is not really good for complex queries.

The figure 5 shows how the number of cach misses depends on the size of the primary data source, with cache of proportional size.

The experiment shows that the behavior of cache depends on its relative size, but not on the absolute size of the primary data source. This might be caused, however, by the uniform distribution of the generated data.

The figure 6 shows how the number of cach misses depends on the ratio of cache size to the primary data source size.

This experiment proves the existence of predictable but still interesting phenomena: at certain intervals of the argument the curve is nearly constant, meaning that at these intevrals an increase of the cache size does not result in any performance gains. In other words, the cache demonstrates a kind of saturation. As soon as the size of cache reached a saturation point, only significant increase of the cache size may provide performance gains. This observation might be practically useful, however, the saturation points depend, of course, on the nature of the data set and the query mix.

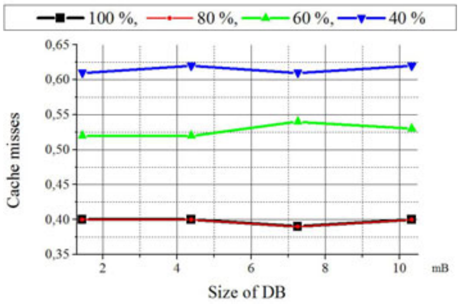


Fig. 5. Size db to cache

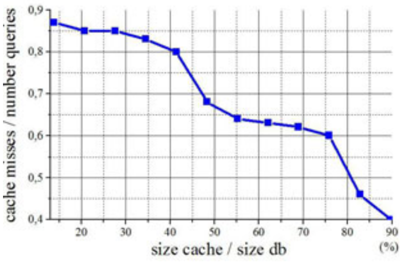


Fig. 6. The saturation

CFDC: A Flash-Aware Buffer Management Algorithm for Database Systems

Yi Ou[1], Theo Härder[1], and Peiquan Jin[2]

[1] University of Kaiserslautern, Germany
{ou,haerder}@cs.uni-kl.de
[2] University of Science & Technology of China
jpq@ustc.edu.cn

Abstract. Classical buffer replacement policies, e.g., LRU, are suboptimal for database systems having flash disks for persistence, because they are not *aware* of the distinguished characteristics of those storage devices. We present CFDC (Clean-First Dirty-Clustered), a flash-aware buffer management algorithm, which emphasizes that clean buffer pages are first considered for replacement and that modified buffer pages are clustered for better spatial locality of page flushes. Our algorithm is complementary to and can be integrated with conventional replacement policies. Our DBMS-based performance studies using both synthetic and real-life OLTP traces reveal that CFDC significantly outperforms previous proposals with a performance gain up to 53%.

1 Introduction

Flash disks will play an increasingly important role for server-side computing, because—compared to magnetic disks—they are much more energy-efficient and they have no mechanical parts and, therefore, hardly any perceptible latency. Typically, flash disks are managed by the operating system as block devices through the same interface types as those to magnetic disks. However, the distinguished performance characteristics of flash disks make it necessary to reconsider the design of DBMSs, for which the I/O performance is critical.

1.1 Performance Characteristics

The most important building blocks of flash disks are flash memory and flash translation layer (FTL). Logical block addresses are mapped by the FTL to varying locations on the physical medium. This mapping is required due to the intrinsic limitations of flash memory [1]. The FTL implementation is device-related and supplied by the disk manufacturer. Many efforts are made to systematically benchmark the performance of flash disks [2,3]. The most important conclusions of these benchmarks are:

– For sequential read-or-write workloads, flash disks often achieve a performance comparable to high-end magnetic disks.

– For random workloads, the performance asymmetry of flash disks and their difference to magnetic disks is significant: random reads are typically two orders of magnitude faster than those on magnetic disks, while random writes on flash disks are often even slower than those on magnetic disks[1].
– Due to the employment of device caches and other optimizations in the FTL, page-level writes with strong *spatial locality* can be served by flash disks more efficiently than write requests without locality. In our context, spatial locality refers to the property of contiguously accessed DB pages being physically stored close to each other.

Interestingly, many benchmarks show that flash disks can handle random writes with larger *request size*s more efficiently. For example, the bandwidth of random writes using units of 128 KB is more than an order of magnitude higher than writing at units of 8 KB. In fact, a write request of, say 128 KB, is internally mapped to 64 *sequential* writes of 2-KB flash pages inside a flash block. Note that sequential access is an extreme case of high spatial locality.

1.2 The Problem

Flash disks are considered an important alternative to magnetic disks. Therefore, we focus here on the problem of buffer management for DBMSs having flash disks as secondary storage. If we denote the sequence of n logical I/O requests $(x_0, x_1, \ldots, x_{n-1})$ as X, a buffer management algorithm A is a function that maps X and a buffer with b pages into a sequence of m physical I/O requests $Y := (y_0, y_1, \ldots, y_{m-1})$, $m \leq n$, i. e., $A(X, b) = Y$.

Let $C(Y)$ denote the accumulated time necessary for a storage device to serve Y, we have $C(Y) = C(A(X, b))$. Given a sequence of logical I/O requests X, a buffer with b pages, and a buffer management algorithm A, we say A is *optimal*, iff for any other algorithm A', $C(A(X, b)) \leq C(A'(X, b))$.

For magnetic disks, $C(Y)$ is often assumed to be linear to $|Y|$. Clearly, this assumption does not hold for flash disks, because C heavily depends on the write/read ratio and the write patterns of Y. Therefore, each I/O request, either logical or physical, has to be represented as a tuple of the form $(op, pageNum)$, where op is either "R" (for a read request) or "W" (for a write request).

While the above formalization defines our problem, our goal is not to find the optimal algorithm in theory, but a practically applicable one that has acceptable runtime overhead and minimizes I/O cost as far as possible.

An intuitive idea to address the write pattern problem is to increase the DB *page size*, which is the unit of data transfer between the buffer layer and the file system (or the raw device directly) in most database systems. It would be an attractive solution if the overall performance could be improved this way, because only a simple adjustment of a single parameter would be required. However, a naive increase of the page size generally leads to more unnecessary I/O (using

[1] As an example, the MTRON MSP-SATA7525 flash disk achieves 12,000 IOPS for random reads and only 130 IOPS for random writes of 4 KB blocks, while high-end magnetic disks typically have 200 IOPS for random I/O [2].

the same buffer size), especially for OLTP workloads, where random accesses dominate. Furthermore, in multi-user environments, large page sizes favor thread contentions. Hence, a more sophisticated solution is needed.

Even with flash disks, maintaining a high hit ratio—the primary goal of conventional buffer algorithms—is still important, because the bandwidth of main memory is at least an order of magnitude higher than the interface bandwidth provided by flash disks. Based on our discussion so far, we summarize the basic principles of flash-aware buffer management as follows, which are also the design principles of the CFDC algorithm:

P1 Minimize the number of physical writes.
P2 Address write patterns to improve the write efficiency.
P3 Keep a relatively high hit ratio.

1.3 Contributions

This paper is an extension of our preliminary work on flash-aware buffer management [4], where the basic ideas of the CFDC algorithm are introduced. The major contributions that distinguish this paper from [4] and improve the CFDC algorithm are:

- We discuss critical issues related to transaction management.
- We demonstrate the flexibility and efficiency of CFDC using a variant of it that integrates LRU-k as a base algorithm. To the best of our knowledge, this is the first work that examines the feasibility of a hybrid algorithm for flash-aware buffer management.
- We introduce metrics to quantitatively study spatial locality of I/O requests.
- We accomplish an extensive empirical performance study covering all relevant algorithms in a DBMS-based environment. So far, such a study is missing in all previous contributions.

The remainder of this paper is organized as follows. Sect. 2 sketches the related work. Sect. 3 introduces our algorithm, while its experimental results are presented in Sect. 4. The concluding remarks are given in Sect. 5.

2 Related Work

LRU and CLOCK [5] are among the most widely-used replacement policies. The latter is functionally identical to the Second Chance [6]: both of them often achieve hit ratios close to those of LRU. LRU-k is a classical algorithm specific for DB buffer management [7]. It maintains a history of page references which keeps track of the recent k references to each buffer page. When an eviction is necessary, it selects the page whose *k-th recent* reference has the oldest timestamp. Parameter k is tunable, for which the value 2 is recommended. Both *recency* and *frequency* of references are considered in its victim selection decisions. Thus theoretically, it can achieve higher hit ratios and is (more) resistant to scans than LRU-based algorithms, for which *recency* is the only concern.

CFLRU [8] is a flash-aware replacement policy for operating systems based on LRU. At the LRU end of its list structure, it maintains a *clean-first region*, where clean pages are always selected as victims over dirty pages. Only when clean pages are not present in the clean-first region, the dirty page at the LRU tail is selected as victim. The size of the clean-first region is determined by a parameter w called the *window size*. By evicting clean pages first, the buffer area for dirty pages is effectively increased—thus, the number of flash writes can be reduced.

LRUWSR [9] is a flash-aware algorithm based on LRU and Second Chance, using only a single list as auxiliary data structure. The idea is to evict clean and cold-dirty pages and keep the hot-dirty pages in buffer as long as possible. When a victim page is needed, it starts search from the LRU end of the list. If a clean page is visited, it will be returned immediately (LRU and clean-first strategy). If a dirty page is visited and is marked "cold", it will be returned; otherwise, it will be marked "cold" (Second Chance) and the search continues.

REF [10] is a flash-aware replacement policy that addresses the pattern of page flushes. It also maintains an LRU list and has a *victim window* at the MRU end of the list, similar to the clean-first region of CFLRU. Victim pages are only selected from the so-called *victim blocks*, which are blocks with the largest numbers of pages in the victim window. From the *set* of victim blocks, pages are evicted in LRU order. When all pages of the victim blocks are evicted, a *linear search* within the victim window is triggered to find a new set of victim blocks. This way, REF ensures that during a certain period of time, the pages evicted are all accommodated by a small number of flash blocks, thus improving the efficiency of FTL.

CFLRU and LRUWSR do not address the problem of write patterns, while REF does not distinguish between the clean and dirty states of pages. To the best of our knowledge, CFDC is the only flash-aware algorithm that applied all the three basic principles P1 to P3 introduced in Sect. 1.

3 The CFDC Algorithm

3.1 The Two-Region Scheme

CFDC manages the buffer in two regions: the *working region* W for keeping *hot* pages that are frequently and recently revisited, and the *priority region* P responsible for optimizing replacement costs by assigning varying priorities to page clusters. A *cluster* is a set of pages located in proximity, i. e., whose page numbers are close to each other. Though page numbers are logical addresses, because of the space allocation in most DBMSs and file systems, the pages in the same cluster have a high probability of being physically neighbored, too. The size of a cluster should correspond, but does not have to be strictly equal to the size of a flash block, thus information about exact flash block boundaries are not required.

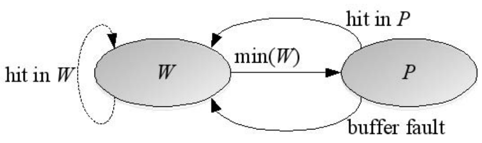


Fig. 1. Page movement in the two-region scheme

A parameter λ, called *priority window*, determines the size ratio of P relative to the total buffer. Therefore, if the buffer has B pages, then P contains λ pages and the remaining $(1 - \lambda) \cdot B$ pages are managed in W. Note W does not have to be bound to a specific replacement policy. Various conventional replacement policies can be used to maintain high hit ratios in W and, therefore, prevent hot pages from entering P.

Fig. 1 illustrates the page movement in our two-region scheme. The code to be executed upon a fix-page request is sketched in Algorithm 1. If a page in W is hit (line 3), the base algorithm of W should adjust its data and structures accordingly. For example, if LRU is the base algorithm, it should move the page that was hit to the MRU end of its list structure. If a page in P is hit (line 5), a page $min(W)$ is determined by W's victim selection policy and moved (demoted) to P, and the hit page is moved (promoted) to W. In case of a buffer fault, the victim is always first selected from P (line 7). Only when all pages in P are fixed, we select the victim from W. Considering recency, the newly fetched page is first promoted to W.

3.2 Priority Region

Priority region P maintains three structures: an LRU list of clean *pages*, a priority queue of *clusters* where dirty pages are accommodated, and a hash table with cluster numbers as keys for efficient cluster lookup. The cluster number is derived by dividing page numbers by a constant *cluster size*.

The victim selection logic in P is shown in Algorithm 2. Clean pages are always selected over dirty pages (line 1–2). If there is no clean page available, a cluster having the lowest priority is selected from the priority queue of dirty pages and the oldest unfixed page in this cluster is selected as victim (line 3–6). The oldest page in the cluster will be evicted first, if it is not re-referenced there. Otherwise, it would have been already promoted to W. Once a victim is selected from a cluster, its priority is set to minimum (line 8) and will not be updated anymore, so that the next victims will still be evicted from this *victim cluster*, resulting in strong spatial locality of page evictions.

For a cluster c with n (in-memory) pages, its priority $P(c)$ is computed according to Formula 1:

$$P(c) = \frac{\sum_{i=1}^{n-1} |p_i - p_{i-1}|}{n^2 \times (globaltime - timestamp(c))} \qquad (1)$$

Algorithm 1. fixPageTwoRegion

```
    data   : buffer $B$ with working region $W$ and priority region $P$;
             page number of the requested page $p$
    result : fix and return the requested page $p$
 1  if $p$ is already in $B$ then
 2      if $p$ is in $W$ then
 3          adjust $W$ as per $W$'s policy;
 4      else
 5          demote $min(W)$, promote $p$;
 6  else
 7      page $q$ := selectVictimPr;
 8      if $q$ is null then
 9          $q$ := select victim from $W$ as per $W$'s policy;
10      if $q$ is dirty then
11          flush $q$;
12      clear $q$ and read content of $p$ from external storage into $q$;
13      $p := q$;
14      if $p$ is in $P$ then
15          demote $min(W)$, promote $p$;
16  fix $p$ in the buffer and return $p$;
```

where $p_0, ..., p_{n-1}$ are the page numbers ordered by their time of entering the cluster. The algorithm tends to assign large clusters a lower priority for two reasons: 1. Flash disks are efficient in writing such clustered pages. 2. The pages in a large cluster have a higher probability of being sequentially accessed.

Both spatial and temporal factors are considered by the priority function. The sum in the dividend in Formula 1, called *inter-page distance* (IPD), is used to distinguish between randomly accessed clusters and sequentially accessed clusters (clusters with only one page are set to 1). We prefer to keep a randomly accessed cluster in the buffer for a longer time than a sequentially accessed cluster. For example, a cluster with pages $\{0, 1, 2, 3\}$ has an IPD of 3, while a cluster with pages $\{7, 5, 4, 6\}$ has an IPD of 5.

The purpose of the time component in Formula 1 is to prevent randomly, but rarely accessed small clusters from staying in the buffer forever. The cluster timestamp $timestamp(c)$ is the value of *globaltime* at the time of its creation. Each time a dirty page is inserted into the priority queue ($min(W)$ is dirty), *globaltime* is incremented by 1. We derive its cluster number and perform a hash lookup using this cluster number. If the cluster does not exist, a new cluster containing this page is created with the current *globaltime* and inserted to the priority queue. Furthermore, it is registered in the hash table. Otherwise, the page is added to the existing cluster and the priority queue is maintained if necessary. If page $min(W)$ is clean, it simply becomes the new MRU node in the clean list.

Algorithm 2. selectVictimPr

```
data   : priority region $P$ consisting of a list of clean pages $L$ in LRU order and
         a priority queue of dirty-page clusters $Q$
result : return a page $v$ as replacement victim
1 if $L$ not empty then
2     $v$ := the first unfixed page starting from the LRU end of $L$;
3 if $v$ is null then
4     cluster $c$ := lowest-priority cluster in $Q$ with unfixed pages;
5     if $c$ not null then
6         $v$ := the oldest unfixed pages in $c$;
7         if $v$ not null then
8             $c.ipd$ := 0;
9 return $v$;
```

After demoting $min(W)$, the page to be promoted, say p, will be removed from P and inserted to W. If p is to be promoted due to a buffer hit, we update its cluster IPD including the timestamp. This will generally increase the cluster priority according to Formula 1 and cause c to stay in the buffer for a longer time. This is desirable since the remaining pages in the cluster will probably be revisited soon due to locality. In contrast, when adding demoted pages to a cluster, the cluster timestamp is not updated.

3.3 Independence of Transaction Management

In principle, recovery demands needed to guarantee ACID behavior for transaction processing [11] may interfere with the optimization objectives P1 – P3. To achieve write avoidance and clustered writes to the maximum possible extent, the buffer manager should not be burdened with conflicting update propagation requirements. Fortunately, our CFDC approach implies a NoForce/Steal policy for the logging&recovery component providing maximum degrees of freedom [11]. NoForce means that pages modified by a transaction do not have to be forced to disk at its commit, but only the redo logs. Steal means that modified pages can be replaced and their contents can be written to disk even when the modifying transaction has not yet committed, provided that the undo logs are written in advance (observing the WAL principle (write ahead log)). Furthermore, log data is buffered and sequentially written—the preferred output operation for flash disks. With these options together, the buffer manager has a great flexibility in its replacement decision, because the latter is decoupled from transaction management. In particular, the replacement of a specific dirty page can be delayed to save physical writes or even advanced, if necessary, to facilitate clustered page flushes and thereby improve the overall write efficiency. Hence, it comes as no surprise that NoForce/Steal is the standard solution for existing DBMSs.

Another aspect of recovery provision is checkpointing to limit redo recovery in case of a system failure, e.g., a crash. To create a checkpoint at a "safe place", earlier solutions flushed all modified buffer pages thereby achieving a transaction-consistent or action-consistent firewall for redo recovery on disk. Such *direct checkpoints* are impractical anymore, because—given large DB buffer sizes—they would repeatedly imply limited responsiveness of the buffer for quite long periods[2]. Today, the method of choice is *fuzzy checkpointing* [12], where only metadata describing the checkpoint is written to the log, but displacement of modified pages is obtained via asynchronous I/O actions not linked to any specific point in time. Clearly, these actions may be triggered to perfectly match with flash requirements and the CFDC principle of performing clustered writes.

3.4 Further Performance Considerations

As outlined, CFDC pe se is not constrained by recovery provisions, in particular, properties such as NoSteal or Force [11]. Such constraints could occur if orthogonality to other components would be violated. An example is the Force policy, with which we could achieve transaction-consistent DB states together with shadowing and page locking. But such an approach would cause low concurrency and overly frequent page propagations—two properties extremely hurting high-performance transaction processing [12].

Prefetching of pages plays an important role for conventional disk-based buffer management: It is not hindered by flash disks. But, because of their random-read performance, prefetching becomes much less important, because pages can be randomly fetched on demand without (hardly) any penalty in the form of access latency. Even better, because prefetching always includes the risk of fetching pages later not needed, CDFC must not use this conventional speed-up technique and can, nevertheless, provide the desired access performance.

As a final remark: The time complexity of our algorithm depends on the complexity of the base algorithm in W and the complexity of the priority queue. The latter is $O(\log m)$, where m is the number of clusters. This should be acceptable since $m \ll \lambda \cdot B$, where $\lambda \cdot B$ is the number of pages in P.

4 Performance Study

4.1 Test Environment

In all experiments, we use a native XML DBMS designed according to the classical *five-layer reference architecture*. For clarity and simplicity, we only focus on its bottom-most two layers, i. e., the *file manager* supporting page-oriented access to the data files, and the *buffer manager* serving page requests. Although designed for XML data management, the processing behavior of these layers is very close to that of a relational DBMS.

[2] While checkpointing often done in intervals of few minutes, systems are restricted to read-only operations. Assume that many GBytes would have to be propagated to multiple disks using random writes (in parallel). Hence, reaction times for update operations could reach a considerable number of seconds or even minutes.

The test machine has an AMD Athlon Dual Core Processor, 512 MB of main memory, is running Ubuntu Linux with kernel version 2.6.24, and is equipped with a magnetic disk and a flash disk, both connected to the SATA interface used by the file system EXT2. Both OS and DB engine are installed on the magnetic disk. The test data (as a DB file) resides on the flash disk which is a 32 GB MTRON MSP-SATA7525 based on NAND flash memory.

We deactivated the file-system prefetching and used *direct I/O* to access the DB file, so that the influences of file system and OS were minimized. All experiments started with a *cold* DB buffer. Except for the native code responsible for direct I/O, the DB engine and the algorithms are completely implemented in Java. CFDC and competitor algorithms are fully integrated into the XML DBMS and work with other components of the DB engine.

In the following, we use CFDC-k to denote the CFDC instance running LRU-k ($k = 2$) and use CFDC-1 for the instance running LRU in its working region. Both of them are referred to as CFDC if there is no need to distinguish. We cross-compared seven buffer algorithms, which can be classified in two groups: the flash-aware algorithms including CFLRU, LRUWSR, REF, CFDC-k, and CFDC-1; the classical algorithms including LRU and LRU-k ($k = 2$). The *block size* parameter of REF, which should correspond to the size of a flash block, was set to 16 pages (DB page size = 8 KB, flash block size = 128 KB). To be comparable, the *cluster size* of CFDC was set to 16 as well. The VB parameter of REF was set to 4, based on the empirical studies of its authors. Furthermore, we used an improved version of CFLRU which is is much more efficient at runtime yet functionally identical to the original algorithm.

4.2 Measuring Spatial Locality

We define the metric *cluster-switch count (CSC)* to quantify the spatial locality of I/O requests. Let $S := (q_0, q_1, \ldots, q_{m-1})$ be a sequence of I/O requests, the metric $CSC(S)$ reflects the spatial locality of S:

$$CSC(S) = \sum_{i=0}^{m-1} \begin{cases} 0, \text{ if } q_{i-1} \text{ exists and in the same cluster as } q_i \\ 1, \text{ otherwise} \end{cases} \qquad (2)$$

Sequential I/O requests are a special case of high spatial locality, where pages are accessed in a forward or reverse order according to their locations on the storage device. If $d(S)$ is the set of distinct clusters addressed by a sequential access pattern S, we have $CSC(S) = |d(S)|$.

Let $R := (p_0, p_1, \ldots, p_{n-1})$ be the sequence of logical I/O requests and S the sequence of physical I/O requests, we further define the metric *cluster-switch factor* (CSF) as:

$$CSF(R, S) = CSC(S)/CSC(R) \qquad (3)$$

CSF reflects the efficiency to perform clustering for the given input R. To compare identical input sequences, it is sufficient to consider the CSC metric alone. For magnetic disks, if we set the cluster size equal to the track size, then $CSC(S)$ approximates the number of disk seeks necessary to serve S. For flash disks, we consider only the CSC and CSF of logical and physical write requests, because flash read performance is independent of spatial locality.

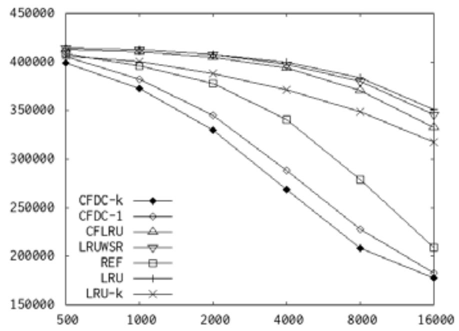
The *clustered writes* of CFDC are write patterns with high spatial locality and thus minimized cluster-switch counts. Compared to CFDC, the sequence of dirty pages evicted by the algorithm REF generally has a much higher CSC, because it selects victim pages from a *set* of victim blocks and the victim blocks can be addressed in any order. Because the sequence of dirty pages evicted can be viewed as multiple sequences of clustered writes that are interleaved with one another, we call the approach of REF *semi-clustered writes.*

4.3 Synthetic Trace

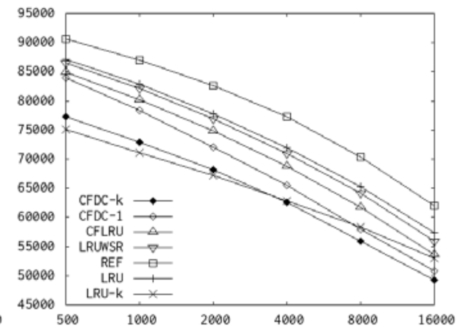
Our synthetic trace simulates typical DB buffer workloads with mixed random and sequential page requests. Four types of page references are contained in the trace: 100,000 single page reads, 100,000 single page updates, 100 scan reads, and 100 scan updates. A *single page read* requests one page at a time, while a *single page update* further updates the requested page. A *scan read* fetches a contiguous sequence of 200 pages, while a *scan update* further updates the requested sequence of pages. The page number of the single page requests are randomly generated between 1 and 100,000 with an 80–20 self-similar distribution. The starting page numbers of the scans are uniformly distributed in $[1, 10^5]$. All the 100,000 pages are pre-allocated in a DB file with a physical size of 784 MB in the file system. Thus a page fault will not cause an extra allocate operation.

We varied the buffer size from 500 to 16,000 pages (or 4–125 MB) and plotted the results of this trace in Fig. 2a. CFDC-k and CFDC-1 are very close, with CFDC-k being slightly better. Both CFDC variants clearly outperform all other algorithms compared. For example, with a buffer of 4,000 page frames, the performance gain of CFDC-k over REF is 26%. Detailed performance break-downs are presented by Fig. 2b, 2c, and 2d, corresponding to the three metrics of interest: number of page flushes, spatial locality of page flushing, and hit ratio. REF suffers from a low hit ratio and a high write count, but is still the third-best in terms of execution times due to its semi-clustered writes. LRU-k performs surprisingly good on flash disks—even better than the flash-aware algorithms CFLRU and LRUWSR. This emphasizes the importance of principle P3.

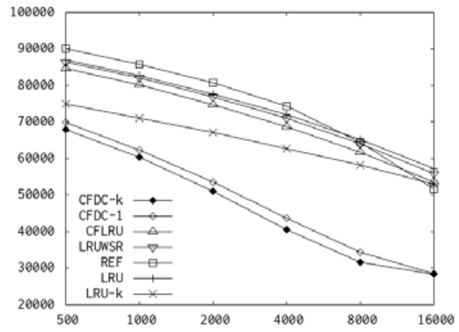
To examine scan resistance, we generated a set of traces by changing the locality of the single page requests of the previous trace to a 90–10 distribution and varying the number of scan reads and scan updates from 200 to 1,600. The starting page numbers of the scans are moved into the interval $[100001, 150000]$. The buffer size configured in this experiment equals the length of a scan (200 pages). Thus, we simulate the situation where sequential page requests push the hot pages out of the buffer. The buffer hits in this experiment are compared in Fig. 3a. While most algorithms suffer from a drop in the number of hits between 5% to 7%, the hits of CFDC-k only decrease by 1% (from 144,926 to 143,285) and those of LRU-k only decrease about 2.5%. This demonstrates that CFDC-k gracefully inherits the merits of LRU-k. Another advantage of CFDC is demonstrated by Fig. 3b: it has always the lowest CSF, i. e., its page flushes are efficiently clustered.



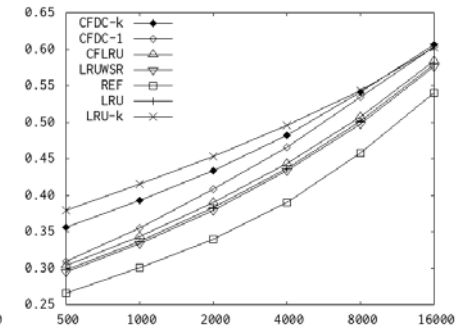
(a) Execution time



(b) Page-flush count

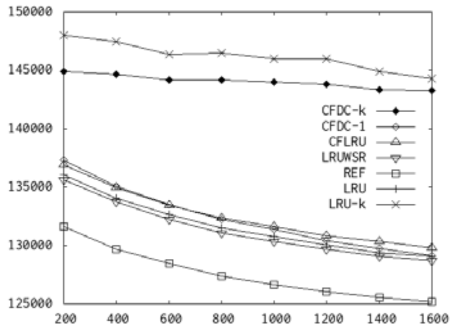


(c) Cluster-switch count

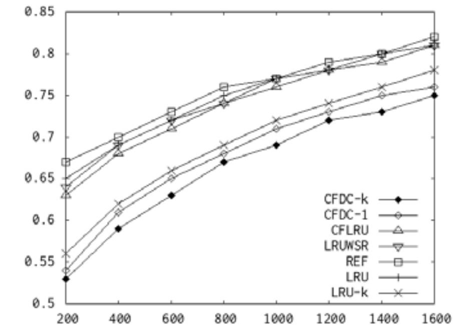


(d) Hit ratio

Fig. 2. Performance figures of the synthetic trace



(a) Hit count



(b) Cluster-switch factor

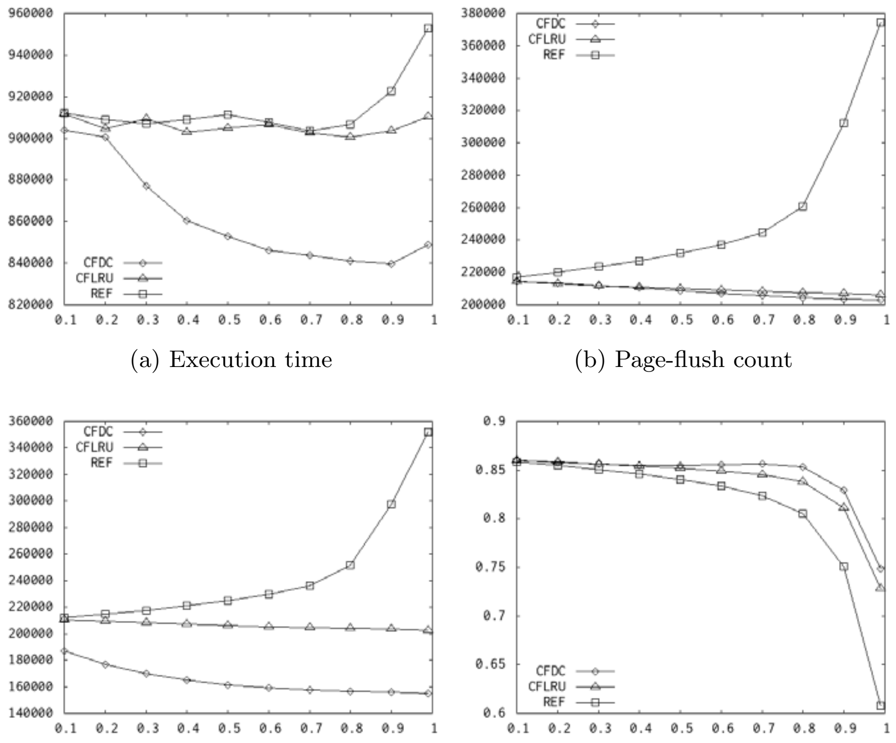
Fig. 3. Increasing the number of scans

4.4 Real-Life OLTP Traces

In this section we present the experiments performed using two real-life OLTP traces. CFDC-k and LRU-k are of lower practical importance due to their higher complexity ($O(\log n)$), therefore, we keep them out for better clarity.

The TPC-C Trace. The first trace was obtained using the PostgreSQL DBMS. Our code integrated into its buffer manager recorded the buffer reference string of a 20-minutes TPC-C workload with a scaling factor of 50 warehouses.

In our two-region scheme, the size of the priority region is configurable with the parameter λ, similar to the parameter *window size* (w) of CFLRU. The algorithm REF has a similar configurable *victim window* as well. For simplicity, we refer to them uniformly with the name "window size". In the experiments discussed so far, this parameter is not tuned—it was set to 0.5 for all related algorithms. To examine its impact under real workload, we ran the TPC-C trace with algorithms CFDC, CFLRU, and REF configured with window size from 0.1 to 0.99 relative to the total buffer size using 1,000 pages in this experiment.



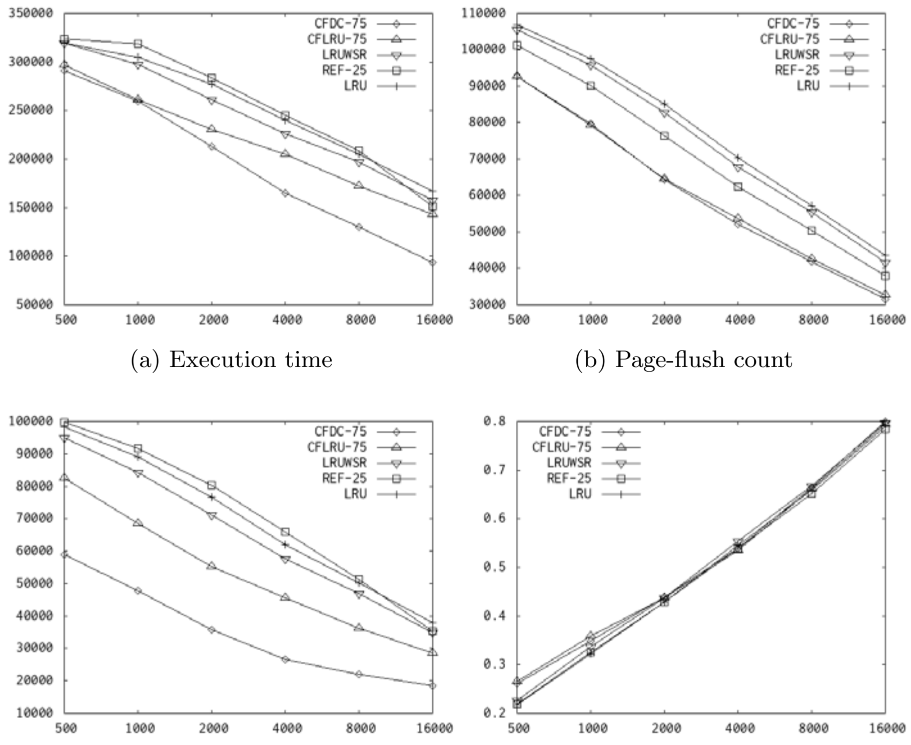
(a) Execution time

(b) Page-flush count

(c) Cluster-switch count

(d) Hit ratio

Fig. 4. Impact of window size on the TPC-C trace



(a) Execution time (b) Page-flush count

(c) Cluster-switch count (d) Hit ratio

Fig. 5. Performance figures of the Bank trace

The performance metrics are shown in Fig. 4. The performance of CFDC benefits from an increasing window size. Its runtime goes slightly up after a certain window size is reached (0.9 in this case). This is because, with the size of the working region approaching zero, the loss of the hit ratio is too significant to be covered by the benefit of reducing physical writes and performing clustered writes in the priority region. Similar behavior is also observed for CFLRU at at window size 0.8. For CFDC and CFLRU, a larger window size leads to smaller number of writes. In contrast, the number of physical writes generated by REF grows quickly with an increase of the window size (Fig. 4b), resulting in a sharp runtime increase beginning at window size 0.8. This is due to two reasons: First, in REF's victim window, the sizes of the blocks are the only concern when selecting a victim block, while temporal factors such as recency and frequency of references are ignored. Second, REF does not distinguish between clean and dirty pages such that an increase of the window size does not necessarily lead to more buffer hits of dirty pages.

The Bank Trace. The second trace used here is a one-hour page reference trace of the production OLTP system of a Bank. It was also used in experiments of [7] and [13]. This trace contains 607,390 references to 8-KB pages in a DB having a size of 22 GB, addressing 51,870 distinct page numbers. About 23% of the references update the page requested, i. e., the workload is read-intensive. Even for the update references, the pages must be first present in the buffer, thus more reads are required. Moreover, this trace exhibits an extremely high access skew, e.g., 40% of the references access only 3% of the DB pages used in the trace [7].

For each of the algorithms CFDC, CFLRU, and REF, we ran all experiments three times with the window size parameter set to 0.25, 0.50, and 0.75 respectively, denoted as REF-25, REF-50, REF-75, etc., and chose the setting that had the best performance. The results are shown in Fig. 5. Even under this read-intensive and highly skewed workload, CFDC is superior to the other algorithms. The performance gain of CFDC over CFLRU is, e.g., 53% for the 16,000-page setting and 33% for the 8,000-page setting. Under such a workload, most of the hot pages are retained in a large-enough buffer. Therefore, the differences in hit ratios become insignificant as the buffer size is beyond 2000 pages.

The performance study does not focus on the MTRON flash disk. We also ran all the experiments on a low-end flash disk (SuperTalent FSD32GC35M) with similar observations. Furthermore, except for the execution time, other metrics collected are independent of any system and device.

5 Conclusions and Outlook

As systematic and empirical study, we extensively evaluated the performance behavior of all competitor algorithms for flash-aware DB buffer management in an identical environment. Therefore, we could accurately and thoroughly cross-compare those algorithms under a variety of parameters and workloads, where the majority of measurement results clearly prove CFDC's superiority among its competitors. The advantages of CFDC can be summarized as follows:

– With the clean-first strategy and the optimization in the priority region, it minimizes the number of physical writes (P1).
– It efficiently writes on flash disks by exploiting spatial locality and performing clustered writes, i. e., it evicts the sequence of writes that can be most efficiently served by flash disks (P2).
– Flash-specific optimizations do not compromise high hit ratios (P3), i. e., a large number of reads and writes can be served without I/O.
– Our two-region scheme makes it easy to integrate CFDC with conventional replacement policies in existing systems. The CFDC-k variant—maybe of lower practical importance due to its higher complexity—served as an example for this flexibility. Modern replacement policies such as ARC [13] and LIRS [14] can be easily integrated into CFDC without modification.

Our experiments did not cover asynchronous page flushing. In practice, page flushes are normally not coupled with the victim replacement process—most of them are performed by background threads. However, these threads can obviously benefit from CFDC's dirty queue, where the dirty pages are already collected and clustered.

In this paper, we explored flash disks as an exclusive alternative to magnetic disks. However, database systems may employ hybrid storage systems, i. e., flash disks and magnetic disks co-exist in a single system. As another option in DBMS I/O architectures, flash memory could serve as a non-volatile caching layer for magnetic disks. Both I/O architectures posing challenging performance problems deserve a thorough consideration in future research work.

Acknowledgement

We are grateful to Gerhard Weikum for providing the OLTP trace and to Sebastian Bächle and Volker Hudlet for the team work. We thank anonymous referees for valuable suggestions that improved this paper. The research is partially supported by the Carl Zeiss Foundation.

References

1. Woodhouse, D.: JFFS: the journalling flash file system. In: The Ottawa Linux Symp. (2001)
2. Gray, J., Fitzgerald, B.: Flash disk opportunity for server applications. ACM Queue 6(4), 18–23 (2008)
3. Bouganim, L., et al.: uFLIP: Understanding flash IO patterns. In: CIDR (2009)
4. Ou, Y., et al.: CFDC: a flash-aware replacement policy for database buffer management. In: DaMoN Workshop, pp. 15–20. ACM, New York (2009)
5. Corbato, F.J.: A paging experiment with the multics system. In: Honor of Philip M. Morse, p. 217. MIT Press, Cambridge (1969)
6. Tanenbaum, A.S.: Operating Systems, Design and Impl. Prentice-Hall, Englewood Cliffs (1987)
7. O'Neil, E.J., et al.: The LRU-K page replacement algorithm for database disk buffering. In: SIGMOD, pp. 297–306 (1993)
8. Park, S., et al.: CFLRU: a replacement algorithm for flash memory. In: CASES, pp. 234–241 (2006)
9. Jung, H., et al.: LRU-WSR: integration of LRU and writes sequence reordering for flash memory. Trans. on Cons. Electr. 54(3), 1215–1223 (2008)
10. Seo, D., Shin, D.: Recently-evicted-first buffer replacement policy for flash storage devices. Trans. on Cons. Electr. 54(3), 1228–1235 (2008)
11. Härder, T., Reuter, A.: Principles of transaction-oriented database recovery. ACM Computing Surveys 15(4), 287–317 (1983)
12. Mohan, C., et al.: ARIES: A transaction recovery method supporting fine-granularity locking and partial rollbacks using write-ahead logging. ACM Trans. Database Syst. 17(1), 94–162 (1992)
13. Megiddo, N., Modha, D.S.: ARC: A self-tuning, low overhead replacement cache. In: FAST. USENIX (2003)
14. Jiang, S., Zhang, X.: LIRS: an efficient low inter-reference recency set replacement policy to improve buffer cache performance. In: SIGMETRICS, pp. 31–42 (2002)

Expert-Assisted Classification Rules Extraction Algorithm

Vili Podgorelec

University of Maribor, FERI, Smetanova ulica 17,
SI-2000 Maribor, Slovenia
Vili.Podgorelec@uni-mb.si

Abstract. Machine learning algorithms nowadays are important and well-accepted tools which help in demanding and ever-more challenging data analysis in many fields. In this paper, we study an approach to machine learning and knowledge discovery, where a learning algorithm uses experts' domain knowledge to induce solutions, and experts use the algorithm and its solutions to enhance their "information processing strength". An adaptation of evolutionary method AREX for automatic extraction of rules is presented that is based on the evolutionary induction of decision trees and automatic programming. The method is evaluated in a case study on a medical dataset. The obtained results are assessed to evaluate the strength and potential of the proposed classification rules extraction algorithm.

Keywords: machine learning, knowledge discovery, classification, medical data.

1 Introduction

The general idea of discovering knowledge in large amounts of data is both appealing and intuitive, but technically it is significantly challenging and difficult, especially in the fields where really huge amounts of relational data have been collected over last decades or even centuries (like medicine). Many such disciplines, generally speaking, are conservative sciences, where all new knowledge has to be fully acknowledged and well understood. In order to fulfill this additional constraint of the knowledge discovery process one should assist domain experts in learning rather than trying to "produce" the knowledge from data fully automatically. Although we will focus on the examples from the field of medicine in the following text, the general idea is very similar also for other fields and can be, therefore, used without constraints.

The literature review reveals a lot of machine learning applications in medicine [6, 7]. Machine learning technology is currently well suited for analyzing medical data, and in particular there is a lot of work done in medical diagnosis in small specialized diagnostic problems. In many cases one or another machine learning algorithm has been developed and/or used for medical diagnosis. In this manner machine learning algorithms are used to automatically learn on the stored data in order to correctly predict, or classify, the unseen cases. Only a few authors suggest the possibility of computers to assist in learning. In [21] authors investigate a simple Bayesian belief network for the diagnosis of breast cancer. In [19] authors present a computer-assisted system for classification of interphase HEp-2 immunofluorescence patterns in autoimmune diagnostics: designed as an assisting system, representative patterns are acquired by an operator with a digital microscope camera and transferred to a personal computer; by use of a novel software package based on image analysis, feature extraction and machine learning algorithms, relevant characteristics describing patterns could be found out. However, both studies still regard the machine learning algorithm as an automated, non-interactive process.

In this paper, we want to present an enhanced approach in machine learning, called machine-assisted learning. In this manner an iterative learning algorithm interacts with an expert and takes advantage of expert's knowledge to learn better and to focus on the preferred patterns. For this purpose, we present a rules induction algorithm that is based on synergetic effects of algorithm's automatic learning and expert's knowledge. Hopefully, this approach should be able to overcome the pitfalls of discovering statistically relevant, but practically worthless or even misleading patterns in data. We are not going to argue that the resulting learning paradigm is necessarily what experts would wish for in their everyday practice, but we are going to suggest that it might be a sound base for what we call the machine-assisted learning.

2 Automatic Classification Rules Extraction Algorithm

For a machine learning system to be useful in solving medical diagnostic tasks, the following features are desired: good performance, the ability to appropriately deal with missing data and with noisy data (errors in data), the transparency of diagnostic knowledge, the ability to explain decisions, and the ability of the algorithm to reduce the number of tests necessary to obtain reliable diagnosis [6].

One of the mostly used methods in medicine are decision trees (DTs) [15], with good efficiency, high classification accuracy and the transparency of the classification process that one can easily interpret, understand and criticize. The usual impurity measures approaches to induction of DTs on the other hand have some disadvantages, like inability to build several trees for the same dataset, inability to use the preferred attributes, etc. Encouraged by the success of evolutionary algorithms for optimization tasks a series of attempts occurred to induce a DT-like models with evolutionary methods [3, 9, 13]. Although GPs have proven extremely difficult to interpret, many researchers have tried to use the power of GAs/GPs for the problem of data mining and knowledge discovery [12, 4]. In this paper we present an upgrade of one such approach to build DTs (or rulesets) with the use of GAs/GPs.

The core method of our approach is Expert-Assisted AREX (Automatic Rules Extractor), an adaptation of a rule extraction algorithm that we have proposed earlier [14]. It is an algorithm that is able to induce a set of classification (if–then) rules based on the given dataset. The two key components are an evolutionary-based construction of classification rules and a multi-level classification approach, combined in a way that enables interaction with experts during the induction process. In this manner, the process would eventually lead to a set of highly accurate rules, which should ideally give an explanation of a problem domain [5, 22].

In order to enable an efficient the interaction with domain experts, the results are presented in a form of classification (if–then) rules, which are straightforward to understand, accept or reject. The input data for the knowledge discovery process is represented with a set of vectors $\vec{o}_1, ..., \vec{o}_N$ – a training set. Each training object $\vec{o}_i$ is described with the values of attributes $a_{i1}, ..., a_{iK}$ and the accompanied decision class ω_i from the set of M possible decisions $[\Omega_1, ..., \Omega_M]$. Attributes can be either numeric (value is a numbers from a continuous interval) or discrete (value is one from the discrete set of possible values). AREX is able to process data with missing values.

Knowledge that is learned with AREX from training data is represented with a set of classification (if–then) rules. Each single rule in a set is in the following form:

```
if <condition> then <decision>,  where
<condition> := <c_1> and ... and <c_d>, and
<decision> := ω, ω ∈ [Ω_1, ..., Ω_M]
```

This means that the rules are 1) simple enough to understand, and 2) powerful enough for an accurate classification.

2.1 Multi-level Gradual Classification Model

One of the most important characteristic of classification rules is their readability to a human [20]. A simple classification rule is easy to comprehend, analyze, and make advantage of. However, for the classification of a whole dataset several rules are needed, each one supporting only a portion of cases. In order for a human to understand the complete classification process, one must not only comprehend each single rule but also their interconnections. In this manner, the complexity of a classification model, represented as a set of rules, greatly depends on the number and the form of all the rules.

In general, real-world medical datasets are very complex, consisting of many attributes, relations, decision classes. It would be irrational to expect high classification accuracy from a small set of simple rules. Usually for the classification of medical datasets complex rulesets, composed of many rules, are needed [10]. On the other hand, in our approach the interpretability of the rules is of the highest importance, as the induced rules should assist experts in learning. In this manner, we decided to build the set of rules gradually, level by level. On a single level only a portion of cases will be classified and the others will be transferred to the next level.

An important contributor to the complexity of rulesets is the diversity of the cases in a dataset. There are usually many so-called special cases, which are quite different from the others. Therefore, the ruleset needs to grow in order to accurately classify all those special cases, which consequentially increases the complexity of the ruleset. If we want to keep the number of classification rules low, those cases should be somehow left out. Afterwards, when common cases are classified with a small set of simple rules, the special cases can be classified separately. Following this proposition, all training objects in AREX are distributed according to their class confusion score (see below, Eq. 2) into several classification levels. For each classification level a specific ruleset is induced. In this manner, the whole dataset is classified gradually (Figure 1).

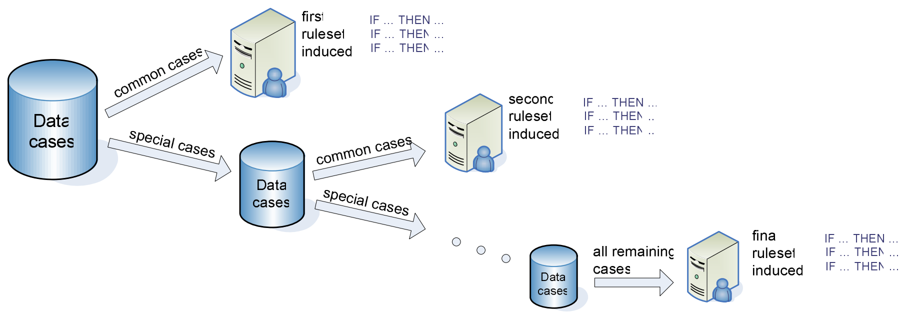


Fig. 1. Multi-level gradual classification approach of AREX

For the distribution of training objects between classification levels an algorithm for the construction of decision trees *genTrees* is used [13]. With *genTrees* several decision trees are induced. For each training object a class classification $cc_i(x)$ is calculated for all decision classes (Eq. 1) – the resulting value represents a number of DTs that classified object x with the decision class i. Then classification confusion score $CCS(x)$ is calculated for each training object x (Eq. 2) – the result represents the confusion score of a set of DTs when classifying object x. If all DTs provide the same classification, then the result is *0*. The higher is $CCS(x)$, the more problematic is the object x for classification. In this manner, the CCS (Eq. 2) actually represents the classification complexity of a specific training object. Based on the classification complexity of an object the appropriate classification level is determined according to the given complexity threshold (class confusion score tolerance).

$$cc_i(x) = \sum_{j=1}^{num\_DTs} \begin{cases} 1; class(DT_j, x) = i \\ 0; otherwise \end{cases} \quad ; \forall i, i \in [1..num\_classes] \tag{1}$$

$$CCS(x) = \sum_{i=1}^{num\_classes} \left(\frac{cc_i(x)}{cc_{\max}(x)} \right)^2 - 1 \tag{2}$$

2.2 The Outline of AREX

AREX is a rules extracting algorithm that induces one classification ruleset for each classification level based on a given training set. It includes a hybrid system of two basic algorithms:

1. an evolutionary algorithm *genTrees* for the construction of DTs [13], and
2. *proGenesys* system for the automatic evolution of programs in an arbitrary programming language.

The algorithm *genTrees* is used for the induction of decision trees, which are then used for the distribution of training objects into classification levels (as described above) and for inducing an initial set of classification rules. The *proGenesys* system is used for the construction of classification rules on a single level. The basic outline of AREX is presented in Figure 2.

```
0. input: a set of training objects S and class
   confusion score tolerance ccst=0.5
1. with genTrees build T evolutionary DTs based on
   training objects from S
2. for each object x from S: if CCS(x)≤ccst, then move
   x from S to S*
3. reject ⌊T/2⌋ of the less accurate DTs and convert the
   remaining ⌈T/2⌉ DTs into R initial classification
   rules
4. with proGenesys create another R classification
   rules randomly; initial population now contains 2×R
   classification rules
5. with proGenesys evolve the final set of
   classification rules based on training set S*
6. find the optimal final set of classification rules
   at the current classification level
7. if S is not empty (there are still non-classified
   training objects for the next level):
       a. add |S| randomly chosen objects from S* to S
       b. increase ccst=2×ccst
       c. repeat the whole procedure for the next
          classification level from step 2
8. finish if S is empty
```

Fig. 2. The basic algorithm AREX

2.3 Evolutionary Algorithm *genTrees* for the Construction of DTs

A first step of the genetic algorithm is the induction of the initial population. A random decision tree is constructed based on the algorithm presented in Figure 3.

```
1. input: number of attribute nodes V that will be in
   the tree
2. select an attribute $A_i$ from the set of all possible K
   attributes and define it a root node tn
3. in accordance with the selected attribute's $A_i$ type
   (discrete or continuous) define a test for this node
   tn:
     a. for continuous attributes in a form of $ftn(A_i) <$
        $\phi_i$ , where $ftn(A_i)$ is the attribute value for a
        data object and $\phi_i$ is a split constant
     b. for discrete attributes two disjunctive sets of
        all possible attribute values are randomly
        defined
```

```
4. connect empty leaves to both new branches from node
   tn
5. randomly select an empty leaf node tn
6. randomly select an attribute A_i from the set of all
   possible K attributes
7. replace the selected leaf node tn with the attribute
   A_i and go to step 3
8. finish when V attribute nodes has been created
```

Fig. 3. The basic algorithm for the construction of a random decision tree

For each empty leaf the following algorithm determines the appropriate decision class: let S be the training set of all training objects N with M possible decision classes ω_1, .., ω_M and N_i is the number of objects within S of a class ω_i. Let S^{tn} be the sample set at node tn (an empty leaf for which we are trying to select a decision class) with N^{tn} objects; N_i^{tn} is the number of objects within S^{tn} of a decision class ω_i. Now we can define a function that measures a potential percentage of correctly classified objects of a class ω_i:

$$F(tn,i) = \frac{N_i^{tn}}{N_i} \tag{3}$$

Decision ω_i^{tn} for the leaf node tn is then marked as the decision ω_i, for which $F(tn,i)$ is maximal. The ranking of an individual DT within a population is based on the local FF:

$$LFF = \sum_{i=1}^{M} w_i \cdot (1 - acc_i) + \sum_{i=1}^{V} c(tn_i) + w_u \cdot nu \tag{4}$$

where M is the number of decision classes, V is the number of attribute nodes in a tree, acc_i is the accuracy of classification of objects of a specific decision class ω_i, w_i is the importance weight for classifying the objects of the decision class ω_i, $c(tn_i)$ is the cost of using the attribute in a node tn_i, nu is number of unused decision (leaf) nodes, i.e. where no object from the training set fall into, and w_u is the weight of the presence of unused decision nodes in a tree.

2.4 System *proGenesys* for Automatic Evolution of Rules

For evolving classification rules we used a system we developed for the evolution of programs in an arbitrary programming language, described with BNF productions – *proGenesys* (program generation based on genetic systems) [11]. In our approach an individual is represented with a syntax tree (a derivation tree). To get the final solution this tree (genotype) is transformed into a program (phenotype) [18]. In the case of rule induction, a single rule is represented as a simple program (i.e. a derivation tree), that evolves through generations.

A good classification rule should simultaneously be clear (most of the objects covered by the rule should fall into the same decision class) and general (it covers many objects – otherwise it tends to be too specific). Those two criteria can be measured using the following equations:

$$generality = \frac{no.\,of\ \ classified\ objects - 1}{no.\,of\ \ objects\ of\ \ decision\ class} \tag{5}$$

$$clearness = 1 - \frac{\frac{\omega_2}{\Omega_2}}{\frac{\omega_1}{\Omega_1}} \tag{6}$$

where ω_1 is the number of objects covered by the rule that belong to the most frequent decision class, ω_2 is the number of objects covered by the rule that belong to the second most frequent decision class, Ω_1 is the number of all objects in the training set that belong to the most frequent decision class of the rule, and Ω_2 is the number of all objects in the training set that belong to the second most frequent class of the rule. Now a fitness function can be defined as

$$FF = clearness \times generality + \sum_{i=1}^{V} c(tn_i) \tag{7}$$

where the last part represents a cost of the use of specific attributes, the same as in the local fitness function *LFF* in building DTs (Eq. 4).

2.5 Finding the Optimal Set of Rules

System *proGenesys* is used to evolve single rules, whereas for the classification of all objects on a specific level a set of rules is required. For this purpose between all the evolved rules a set of rules should be found that together classify all the objects – with high classification accuracy and a small number of rules. A problem is solved with a simple genetic algorithm that optimizes the following fitness function:

$$FF = \sum_{i=1}^{M} w_i \cdot (1 - acc_i) + \sum_{i=1}^{V} c(tn_i) + w_m \cdot nm + w_u \cdot nu \tag{8}$$

where the first two parts are the same as in the LFF for building decision trees (Eq. 4), and instead of a penalty for unused decision nodes penalties for multiple classified objects and non-classified objects are used here. The appropriate coverage of the training set is thus achieved by reducing the number of not classified and multiple classified objects [2].

3 A Case Study

In order to assess the presented approach, the method has been applied to a medical dataset. The early and accurate identification of cardiovascular problems in children patients is of vital importance. Based on a specially defined protocol to collect the important data, a cardiovascular dataset has been composed from the clinical database to be used for the knowledge discovery process. It contains data for 100 pediatric patients from Maribor Hospital. The attributes include general data (age, sex, etc.), a health status (data from family history and child's previous illnesses), a general cardiovascular data (blood pressure, pulse, chest pain, etc.) and more specialized cardiovascular data – data from child's cardiac history and clinical examinations (with findings of ultrasound, ECG, etc.) [1]. In this manner 37 different values have been determined for each patient, which should reveal the type of patients' cardiovascular problems.

In the dataset each patient has been diagnosed in one of the five different possible diagnosis (Table 1). The distribution of diagnosis classes is non-uniform, with the frequency of the most frequent class 5 times higher than the frequency of the less frequent class.

Table 1. The basic information for the used dataset

decision class	diagnosis	
	name	*frequency*
class 0	innocent heart murmur	36
class 1	congenital heart disease with left to right shunt	12
class 2	aortic valve disease with aorta coarctation	23
class 3	arrhythmias	22
class 4	non-specific chest pain	7

The small size of the dataset, the high number of decision classes and the non-uniformity of class distribution all speak for the difficult classification problem. On the other hand, the number of attributes and the lack of comprehensive clinical trials in the specific domain, make the problem very interesting for the knowledge discovery process.

3.1 Results

The most common measure of efficiency when assessing a classification method is accuracy, defined as a percentage of correctly classified objects from all objects (correctly classified and not correctly classified). In many cases, especially in medicine, accuracy of each specific decision class is even more important than the overall accuracy. When there are two decision classes possible (i.e. positive and negative patients), the common measures in medicine are sensitivity and specificity. In our case of cardiovascular dataset, as in many other real-world medical datasets, there are more than two decision classes possible, actually five. In this case the direct calculation of sensitivity and specificity is not possible. Therefore, the separate class accuracy of *i*-th single decision class is calculated as:

$$ACC_{k,i} = \frac{T_i}{T_i + F_i} \tag{9}$$

and the average accuracy over all decision classes is calculated as

$$ACC_k = \frac{1}{M} \cdot \sum_{i=1}^{M} \frac{T_i}{T_i + F_i} \qquad (10)$$

where M represents the number of decision classes.

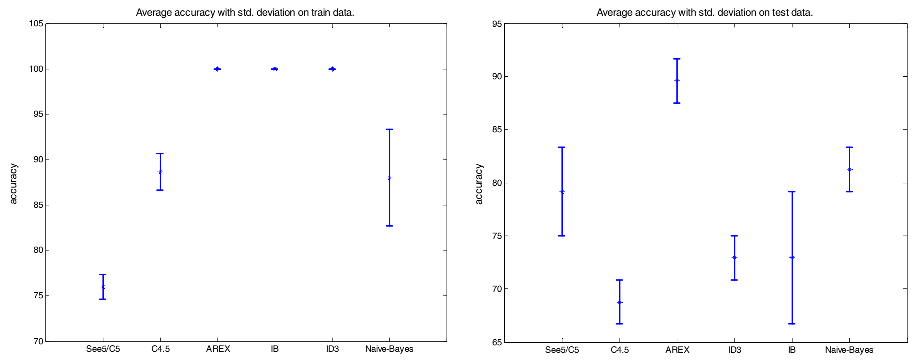
Quantitative results. To evaluate the classification results of our method AREX the above mentioned measures of efficiency were calculated based on 10 independent evolutionary runs. For the comparison with other classification algorithms we calculated the classification results obtained with algorithms ID3, C4.5, See5/C5 [16, 17], Naïve-Bayes (N-B), and instance-based classifier (IB) [8]. Based on 10-fold cross validation for each algorithm we calculated overall accuracy on a training set and a testing set (Table 2 and Figure 4), average class accuracy on a training set and a testing set (Table 3 and Figure 5), and also average accuracies on specific classes on a testing set (Figure 6).

From the classification results (Table 2 and Figure 4) it can be seen that there are three algorithms that achieved perfect score on a training set, namely ID3, IB, and AREX. On the other hand, no algorithm scored perfectly on a testing set. The best score was achieved by AREX (89.85), followed by Naïve-Bayes (81.25) and See5/C5 (79.17). Both ID3 and IB, which scored perfectly on a training set, achieved much worse results – this fact speaks for extreme over-fitting in those two (also present in C4.5 to some extent). The best classification results of AREX on both training and testing set speak for high learning and generalization capability of AREX (at least for the used dataset). Beside AREX, only See5/C5 and Naïve-Bayes scored similarly on both training and testing set (although not as well as AREX). Standard deviation of the results is also the lowest with AREX (beside C4.5, ID3, and Naïve-Bayes), what is actually surprisingly outstanding for an evolutionary method.

When considering the average class accuracy, i.e. accuracies achieved on single decision classes (Table 3 and Figure 5), similar conclusions as with the overall accuracy can be made regarding the comparison of a training and a testing set scores. The best result (by far) on a testing set was achieved by AREX (85.83), followed by Naïve-Bayes and See5/C5 (71.67). Again, AREX achieved the lowest standard deviation of the results. This result is very important and speaks for AREX's ability to

Table 2. Average classification accuracy on training and on testing set for different classification algorithms

classification	accuracy on the training set [%]		accuracy on the testing set [%]	
algorithm	*average*	*stdev*	*average*	*stdev*
See5/C5.0	76.00	1.33	79.17	4.17
C4.5	88.67	2.00	68.75	2.08
AREX	100.00	0.00	89.58	2.08
ID3	100.00	0.00	72.92	2.08
IB	100.00	0.00	72.92	6.25
Naïve-Bayes	88.00	5.33	81.25	2.08

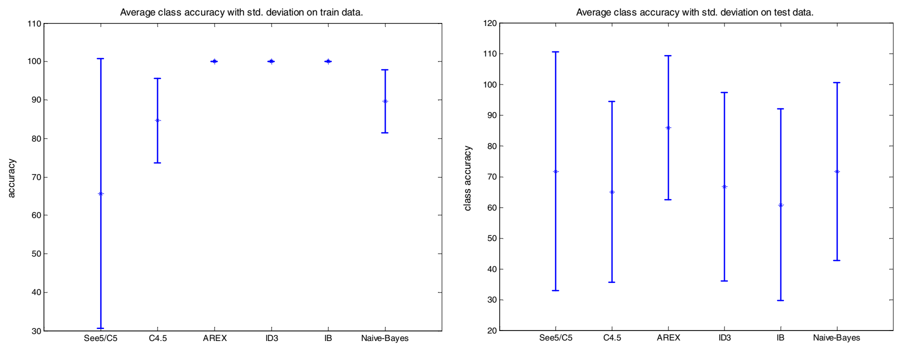


a) training data b) testing data

Fig. 4. Average accuracy for different classification algorithms

Table 3. Average class accuracy on training set and on testing set for different classification algorithms

classification algorithm	class accuracy on the training set [%]		class accuracy on the testing set [%]	
	average	*stdev*	*average*	*stdev*
See5/C5.0	65.67	35.00	71.67	38.80
C4.5	84.65	11.01	65.00	29.30
AREX	100.00	0.00	85.83	23.33
ID3	100.00	0.00	66.67	30.61
IB	100.00	0.00	60.83	31.17
Naïve-Bayes	89.64	8.12	71.67	28.93



a) training data b) testing data

Fig. 5. Average class accuracy for different classification algorithms

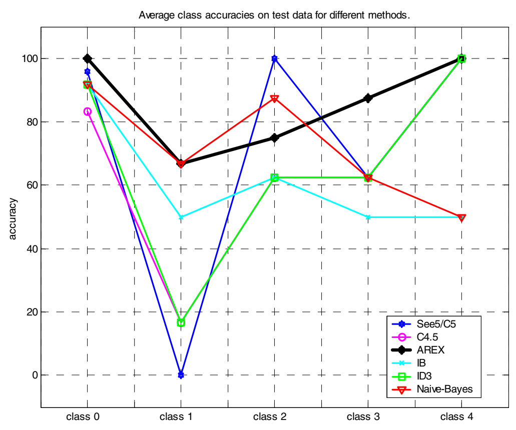


Fig. 6. Average class accuracies on a testing set for different classification algorithms

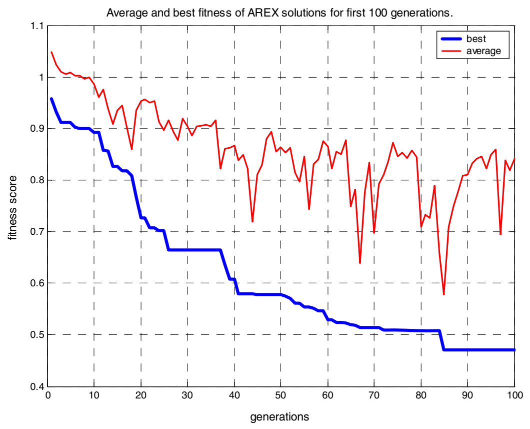


Fig. 7. Change of fitness through the first 100 generations

classify equally well different decision classes, either if they are more or less frequent. In most of the classification algorithms the less frequent decision classes are often neglected or even ignored, because they do not contribute much to the overall accuracy. However, especially in medicine those less frequent classes are very important, since they usually represent specific patients, which have to be treated even with more care. Furthermore, the correct classification of those rare cases has the highest potential to reveal some new patterns to medical experts. The class accuracies for all five decision classes, as achieved with different algorithms, are presented in Figure 6. There can be seen that AREX scored the best in all but class2.

For the clear understanding of induced rulesets by domain experts it is important that the number of classification rules is not very high. When using evolutionary methods it is also important to guarantee the convergence of evolutionary runs with stable increase of fitness scores. Figure 7 shows an average convergence rate of an

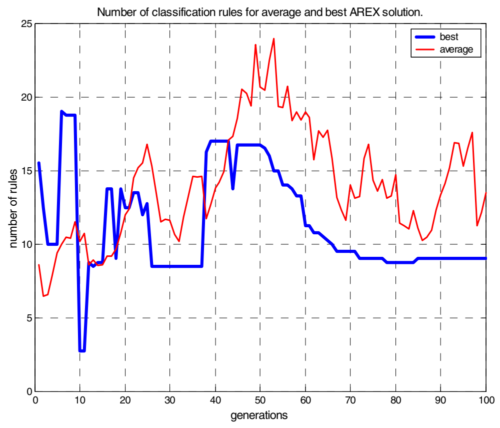


Fig. 8. Number of induced classification rules through the first 100 generations

evolutionary run for the first 100 generations; a moderate and stable increase can be seen with a few bigger drops. Figure 8 shows the number of induced classification rules for average and best solution during the evolution of the first 100 generations; after some initial oscillation the number of rules remains stable between 5 and 15 rules.

4 Conclusion

The quantitative classification results of the presented method on the studied cardiovascular dataset are outstanding; regarding the accuracy and the class accuracy our method outperformed all of the well known classification algorithms considered in the case study. Although the experiments do not directly show the influence of expert knowledge on the classification results, it is reasonable to believe that it had contributed a small, but important part.

On the other hand, it is exactly this influence of expert knowledge on the classification process that can be seen as the biggest disadvantage of the proposed method. Namely, the machine-assisted learning paradigm of AREX requires an expert to qualitatively assess a set of rules for their relevancy several times before the whole process is finished. None of the compared algorithms requires any interaction, which makes them more appropriate for everyday practice. On the other hand, it is enough for the knowledge discovery process to be performed only once, after which the obtained results can be used indefinitely. Especially if they are good.

Considering all the results, it can be concluded that the presented machine-assisted learning approach, as presented in this paper and implemented in AREX, equip the medical experts with a powerful tool to 1) help them in diagnosing, 2) confirm their existing knowledge about a medical problem, and 3) enable them searching for new facts, which should reveal some new interesting patterns and possibly improve the existing medical domain knowledge.

In this manner, we would like to perform more exhaustive testing on other datasets and in other domains. Furthermore, we will work on the integration of domain knowledge right within the machine learning process in order to reduce the experts' effort in assessing the evolving classification rules.

References

1. Anderson, H.R., et al.: Clinicians Illustrated Dictionary of Cardiology. Science Press, London (1991)
2. Anglano, C., Giordana, A., Lo Bello, G., Saitta, L.: An experimental evaluation of coevolutive concept learning. In: Proceedings of the International Conference on Machine Learning ICML 1998, pp. 19–27 (1998)
3. DeJong, K.A.: Hybrid learning using genetic algorithms and decision trees for pattern classification. In: Proceedings of the IJCAI Conference (1995)
4. Freitas, A.A.: A survey of evolutionary algorithms for data mining and knowledge discovery. In: Advances in Evolutionary Computation, pp. 819–845. Springer, Heidelberg (2002)
5. Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Morgan Kaufmann, San Francisco (2000)
6. Kononenko, I.: Machine learning for medical diagnosis: history, state of the art and perspective. Artificial Intelligence in Medicine 23, 89–109 (2001)
7. Magoulas, G.D., Prentza, A.: Machine Learning in Medical Applications. In: Paliouras, G., Karkaletsis, V., Spyropoulos, C.D. (eds.) ACAI 1999. LNCS (LNAI), vol. 2049, pp. 300–307. Springer, Heidelberg (2001)
8. Machine Learning in C++, MLC++ library, `http://www.sgi.com/tech/mlc`
9. Papagelis, A., Kalles, D.: Breeding decision trees using evolutionary techniques. In: Proceedings of the ICML 2001 (2001)
10. Peña-Reyes, C.A., Sipper, M.: Evolutionary computation in medicine: an overview. Artificial Intelligence in Medicine 19(1), 1–23 (2000)
11. Podgorelec, V.: ProGenesys – program generation tool based on genetic systems. In: Proceedings of the ICAI 1999, pp. 299–302 (1999)
12. Podgorelec, V., Kokol, P.: Towards more optimal medical diagnosing with evolutionary algorithms. Journal of Medical Systems 25(3), 195–220 (2001)
13. Podgorelec, V., Kokol, P.: Evolutionary induced decision trees for dangerous software modules prediction. Information Processing Letters 82(1), 31–38 (2002)
14. Podgorelec, V., Kokol, P., Molan Stiglic, M., Hericko, M., Rozman, I.: Knowledge discovery with classification rules in a cardiovascular dataset. Computer Methods and Programs in Biomedicine 80(1), S39–S49 (2005)
15. Podgorelec, V., Zorman, M.: Decision Trees. In: Meyers, R.A. (ed.) Encyclopedia of Complexity and Systems Science, vol. 2, pp. 1826–1845. Springer, New York (2009)
16. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann, San Francisco (1993)
17. RuleQuest Research Data Mining Tools, `http://www.rulequest.com`
18. Ryan, C.: Shades – a polygenic inheritance scheme. In: Proceedings of Mendel 1997 Conference, pp. 140–147 (1997)
19. Sack, U., Knoechner, S., Warschkau, H., Pigla, U., Emmrich, F., Kamprada, M.: Computer-assisted classification of HEp-2 immunofluorescence patterns in autoimmune diagnostics. Autoimmunity Reviews 2, 298–304 (2003)
20. Tan, K.C., Yu, Q., Heng, C.M., Lee, T.H.: Evolutionary computing for knowledge discovery in medical diagnosis. Artificial Intelligence in Medicine 27(2), 129–154 (2003)
21. Wang, X.-H., Zheng, B., Good, W.F., King, J.L., Chang, Y.-H.: Computer-assisted diagnosis of breast cancer using a data-driven Bayesian belief network. International Journal of Medical Informatics 54, 115–126 (1999)
22. Weiss, S.M., Indurkhya, N.: Predictive Data Mining. Morgan Kaufmann, San Francisco (1998)

Information Extraction from Concise Passages of Natural Language Sources

Sandi Pohorec, Mateja Verlič, and Milan Zorman

University of Maribor, Faculty of Electrical Engineering and Computer Science, Smetanova ulica 17, 2000Maribor, Slovenia
{Sandi.Pohorec,Mateja.Verlic,Milan.Zorman}@uni-mb.si

Abstract. This paper will present a semi-automated approach for information extraction for ontology construction. The sources used are short news extracts syndicated online. These are used because they contain short passages which provide information in a concise and precise manner. The shortness of the passage significantly reduces the problems of word sense disambiguation. The main goal of knowledge extraction is a semi-automated approach to ontology construction.

Keywords: knowledge engineering, knowledge formalization, knowledge extraction, ontology construction, ontology population, natural language processing, POS tagging, named entity recognition.

1 Introduction

Information extraction (IE) is a technology based on analysing natural language in order to extract snippets of information. The process takes texts (and sometimes speech) as input and produces fixed format,unambiguous data as output [1]. Information extraction tries to extract, from natural language, several different types of information: entities, mentions, descriptions of entities, relations between entities and events involving entities. Entities in text are people, places, dates, mentions are the places in the text where an entity is mentioned (by its name or reffered to). The process of named entity recognition is believed to be *weakly domain-independent*. This means that entity recognition would need minor changes when switching the process from healt news to technology news and major changes when switching from news to scientific literature [1].

Ontology construction is defined [1] as consisting of the following steps: domain understanding, data understanding, ontology learning, ontology evaluation and refinement. The task of ontology learning is addressed in this paper. We are focusing on the definition of key ontology terms and relations between them. As the example we will use an ontology that focuses on “computer hardware” and we will be using the news items to find new instances of ontology concepts and additional information about them. For these we are using knowledge discovery approaches on a collection of documents. The approach is semi-automated, human interaction is intensive in the first phases. We are using natural language processing in order to determine domain terms and information about them. This is the data used to populate the ontology (or enhance it). The understanding of the text meaning is a decisive element in the extraction process.

To understand natural language a complex process is required. Meaning cannot be inferred with only the simple sequencing of individual word meaning from dictionaries. The process of understanding natural language is comprised of sentence parsing, semantic meaning representation and interpretation of the meaning in the context of the target domain. Naturally as the average length of the processed texts increases, this also increases the complexity of the sense disambiguation. For these reasons we have chosen to process shorter, more concise passages of text where the meaning is summarized with only a few sentences. Since there is a limited amount of sources of these particular data type we have chosen the online news services as the primary data source. Also an important factor in the natural language understanding is the scope of the approach. In other words the factor varies with regard to the understanding of general (multi-domain) text versus understanding domain specific text. Since we are extracting only domain-specific data we are assigning only domain-specific meaning to the terms extracted.

2 Domain and Data Understanding

The process of domain understanding is vital to the success of knowledge extraction. A knowledge engineer rarely possesses enough knowledge on the target domain. One of the first tasks in the domain understanding is the definition of the domain boundaries. This is accomplished with an extensive examination of the target domain. Understanding of the domain is crucial if the knowledge engineer is to incorporate his skills in the entire knowledge extraction process. Domain boundaries are defined with a firm, formal definition of the target domain. This definition should leave no questions regarding what lies inside the domain and what outside. For our purposes the domain is defined as knowledge on computer hardware. The source data for the extraction process will be online news. They are syndicated with the use of RSS (Really Simple Syndication) format. RSS [2] is a web content syndication format; all RSS files must conform to the XML 1.0 [3] specification. According to the RSS each document is a *channel*. The channel has a list of items. The required elements of a channel are: title, link and description. Elements of an item are: title, link, description, author, category, comments, enclosure, guid, publication date and source.

In order to gather the data we have implemented a crawler similar to the web crawlers of search engines [4].The major tasks of the crawler we developed, for the transfer of news sources, do not differentiate from a typical web crawler: continuous visiting of content and content downloading. However the content in the news syndicated is for the most part already in plaintext. Therefore the problem of the transformation to plaintext that in traditional crawlers is quite demanding is reduced only to the transformation of the content in html.

At the first stage of the news gathering all news sources are treated equally - the revisiting policy is uniform and the revisiting interval is set to a very frequent degree (so not to miss any news).After some time when the statistics on the frequency of news publishing for individual sources are calculated the policy changes to a proportional one. The latter dictates frequency of visits according to the publishing style of individual sources. The sources that publish more visits are visited more frequently and others less frequently. The interval between visits is based on the probability of observed content change [5]. The metric is calculated with the following equation:

$$Prob\ (content\ change) = 1 - e^{-RT} \tag{1}$$

This way the work load is more evenly distributed and the number of visits which result in no new content is reduced.

The crawler in the terms of content limitation is a focused crawler [6].

3 Natural Language Processing

In order to acquire new terms, their descriptions and relations to other entities, multiple steps of natural language preprocessing are necessary. For the approach we are presenting these include: tokenization, part-of-speech tagging, lemmatization, and chunking (grouping of individual words into syntactically linked parts).

For the entire natural language processing process (NLP) we are using a custom framework that provides dynamic workflow capabilities for NLP. The system is designed with formal, strong typed modules that can be combined to a custom workflow. Each module is a NLP module that is formally described (formal signature of the required input parameters and type of results returned) and enables automatic validation of the workflow (if one module can be used with another). The architecture of the framework is shown on Fig. 1.For the news processing we have created a custom workflow that is comprised of the following modules: word and sentence level tokenization, part-of-speech (POS) tagging and chunking, named entity recognition and the sentence level syntactic features resolution (relations between individual words in a sentence). These modules are examined in their corresponding subsections.

3.1 Word and Sentence Level Tokenization

When processing natural language the texts are essentially just sequences of characters without explicit annotations of word and sentence level boundaries. Since most NLP processing applications work on sentences and words, they require appropriate levels of tokenization [7].The process of tokenization is highly dependent on the language on which it is being done. Alphabet languages separate words with spaces. This enables the simplest of approaches (use space as the delimiter between words) to achieve effective word tokenization. This level, although quite capable, is insufficient for the use of tokenization in a knowledge extraction process. Well known problems in tokenization are: ambiguity of the *period* punctuation mark and others (*exclamations*), multi word expressions (for instance 1. January 2010), abbreviations where one word actually represents two (“we’ll” actually represents “we will”), separated words (two words are actually one) and missing spaces (typing errors).

Punctuation ambiguity is most common with the *period* mark. The period can be used for in an abbreviation, an acronym, a date and other. Other punctuations are less ambiguous. Of course there are exceptions, such as “Yahoo!” where the exclamation point is a part of the company name. Colon and semicolon are often difficult to

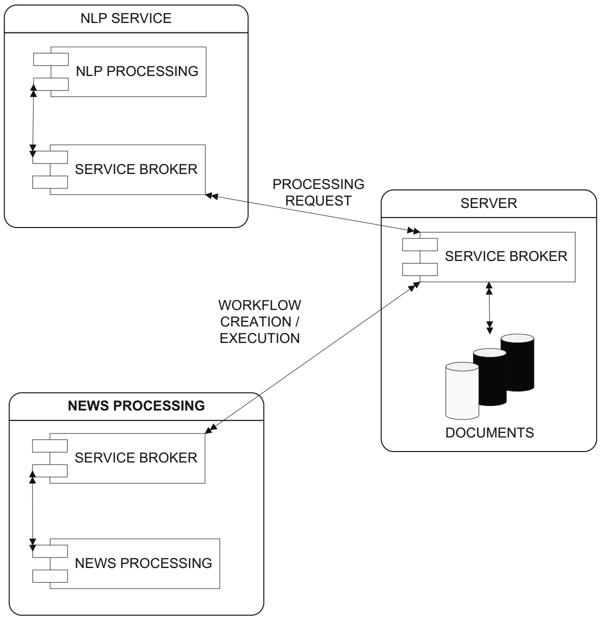


Fig. 1. The architecture of the framework used for natural language processing. The consumer application (news processing) uses the server service broker to create and execute workflows that combine different NLP services. The server service broker sequentially invokes each NLP service until the workflow is completed.

interpret since they can represent the end of a sentence, the beginning of an enumeration or just punctuations within the sentence. Also multi word expressions are an exception to the rule that a space separates two words. These expressions should be handled without separating the words. Separated words (two words that are actually one, they contain a hyphen and a carriage return line feed character; CRLF) that are a consequence of line breaks are most commonly an instance of one of the following cases: the word was split because of the line break (the hyphen and the CRLF should be removed), the word is normally written with a hyphen (the CRLF should be removed) and the passage contains a phrase such as pre- and post-processing (the CRLF should be replaced with a space).

Since the tokenization process is highly content dependent it is essential to increase its effectiveness with an analysis of the individual words in order to determine which punctuation marks are used as delimiter and which only serve as syntactic or semantic annotations (period ending an abbreviation is not a delimiter, it is however an syntactic annotation). For the purpose of analysis and examination the tagger that combines hard coded rules and an ensemble of specialized taggers (also used with POS tagging). These are combined with the *boosting*[8] method. The taggers ensemble is comprised of four different classes of taggers: (1) an example based tagger (with example dataset), (2) a tagger that generates examples according to some logic,(3) a tagger that uses regular expressions and (4) a combination tagger that tags a sequence of words tagged by other taggers. Examples of taggers implemented are:

name tagger (type 1), number generation tagger (type 2 - generates word representation for numbers), email and web address taggers (type 3) and multiword entity tagger (type 4). The latter tags entities such as "prof. John Ryan" where a combination of an abbreviation and two names forms a multiword entity.

3.2 Part-of-Speech Tagging and Chunking

For the part of speech tagging, we are using a hybrid tagger composed of the TnT tagger [9] and our own dictionary-pattern based tagger. The combination is used for mutual verification of the correctness of tagging and enhancing the results. Our implementation of the tagger is a pattern based approach that is intended for sentence scale tagging. The algorithm of the pattern based tagger follows these steps:

1. Loop through all words in a sentence. Try to determine each word's part-of-speech with the use of dictionaries. If there is no dictionary entry, tag the current word with unknown tag ("*?*"). If the word has multiple parts-of-speech (the same word can be a noun or a verb) tags assign a tag that indicates multiple possible *MSD* (morphological syntactic descriptor) tags and associate all of the possible *MSD* tags to the word. If however the word has only one possible *MSD* tags assign that tag to the word.
2. Check if any of the unknown words (tagged with "*?*") can be POS tagged with the ensemble of specialized taggers (use only taggers which are known to be highly reliable).
3. If the sentence contains any words with multiple possible tags: find sentences in the corpora that are similar to the currently processed sentence and try to resolve the words with multiple tags. Use statistical similarity measures to determine which of the possible tags the correct one is.
4. If the sentence still contains (after steps 1 and 2) any unknown words: determine the possible word tags from the similar sentences in the corpora. Evaluate the statistical probabilities of each possibility and use the most probable one.

Step 1 is rather simple to understand, since it is mainly concerned with dictionary lookup. The ensemble used in step 2 is the same briefly described in subsection 3.1. So we continue with the explanation of step 3. Each of the sentences in the example set has the full string representation, each of the words have MSD tags assigned. Each sentence also has a sentence pattern. The pattern is a single string that contains only non-numeric characters. Each word in the sentence is represented as a single character (the first letter in the MSD). The "NVSN" pattern describes a sentence where the first word is a noun (N), the second a verb (V), the third a preposition (S) and the fourth again a noun (N). The search for similar sentences is conducted with the use of full-text search operators in a relational database. This is the reason why unknown words are tagged with "?". "?" is a wild card operator (any character match). Using the wildcard operator it is simple to find sentences that are similar to the current one. The search itself is executed in two phases: in the first the sentences that match the number of words are sought and in the second sentences that exceed the current word count are sought (they are longer and contain a sequence of words similar to the current sentence).

Step 4 evaluates the statistical probability that a certain word can be assigned a specific part-of-speech tag. This process is a combination of statistical factors and suggestions provided by the specialized taggers ensemble. The statistical factors are essentially the matching percentage between the processed sentence and the pattern matched one. If every word, except the unknown one, matches in number, gender, case, etc. it is very probable that the unknown word is in the same part-of-speech as the word in the pattern matched sentence. The statistical factors are combined with the ensemble suggestions according to the following possible scenarios: (i) statistics suggests an MSD that matches the suggestion from the ensemble: the MSD is accepted and used to tag the unknown word, (ii) the suggestions do not match: the choice between them is made according to the probability and the known reliability of the ensemble member that provided the suggestion (if multiple members suggested it, the reliability is a normalized sum). If the calculated probability is low and the taggers are reliable (have been proven in the past) the ensemble suggestion is selected. In the opposite case the tag from the statistical evaluation is selected.

When every word has been POS tagged, they are chunked together according to grammar rules. The result of this process is the *parse tree*. An example of the tree for a simple sentence is shown on Fig.2. At the end of this phase of preprocessing, words have been assigned their individual *local information* (POS tag and chunk identification), necessary for the sentence disambiguation process.

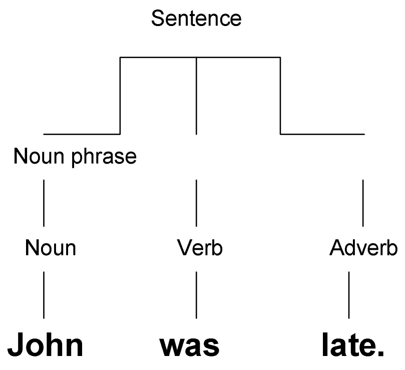


Fig. 2. The result of the POS tagging and chunking, the parse trees, shown on the example sentence "John was late". Each word is tagged with a POS tag and the words “John” and “was” are chunked together to form a noun phrase.

3.3 Named Entity Recognition

For the named entity recognition we are again using a hybrid approach. For the detection of known entities a dictionary of all ontology terms is used. As for the detection of previously unknown entities we are using a trend analysis based approach. It incorporates time aligned multi language news on the same subject. This approach is based on the common news property: when something occurs it is quickly reported all over the world. Usually there is a time interval, on which news on the new subjects are reported continuously. So when a new term emerges it is mentioned constantly for a (short) period of time. This metric, we believe, can be used to distinguish ontology terms from other, common words.

The candidates for new ontology terms are the unknown words in the processed news.Since named entities are sought after only unknown words that have been tagged as nouns by the POS tagger are considered for further evaluation.

The approach for the verification of an individual unknown word as a domain term is based on trends analysis. A prerequisite for the approach is a long enough history of processed news stories. The analysis of every word is essentially the daily word occurrence frequency analysis. The procedure is comprised of these steps:

- Determine the daily occurrence frequencies for the word for a predetermined time period.
- Extract the individual intervals where the word frequencies are higher from the neighboring days.
- Check if the intervals are spikes (the average frequency count in the interval is much higher than its surroundings).
- Cross check the word usage in other languages (news items in other languages).

Fig. 3 shows the chart of the frequency of occurrence for the term “iPad” on a time period from mid-2008 until May of 2010. As we can see this term does have the characteristic spikes on its usage counters. The first spike occurs on the interval from the 25. to the 27. of May 2009. While the previous interval (April 2008 – 25. May 2009) has an average frequency of occurrence of 3, the three days in question have an average of 138. Similar, only more powerful in frequency of occurrence, is the interval from the 30. of March 2010 to the 14. of April where an average frequency of the term is 1318. The spikes on the English language news items are also correlated to the non-English news items therefore proving this to be a domain term.

The local features of the word, confirmed by the analysis to be domain terms, are enhanced with the tag indicating this word to be a domain term.

3.4 Sentence Level Syntactic Features

The syntactic features of the sentences need to be inferred in order to be able to formalize the meaning of the sentence. The formalization process transforms the meaning of the sentence from a natural language notation to a formal notation. Since the purpose is to construct ontology on the target domain, the formal notation is that of ontology.

Ontology is formally defined [10] as a structure:

$$O = (C, T, R, A, I, V, \leq c, \leq t, \sigma R, \sigma A, \rho C, \rho T, \rho R, \rho A)$$

It consists of disjoint set of: (C) concepts, (T) types, (R) relations, (A) attributes, (I) instances and (V) values. The partial orders $\leq c$ (on C) and $\leq t$ (on T) define a concept hierarchy and a type hierarchy, respectively. The function $\sigma R: R \rightarrow C^2$specifies which concepts can be linked by this relation. Function $\sigma A: C \times T$ lists the concepts that an individual attribute belongs to and its datatype. The partial instantiation functions are: instances to concepts (ρC), values to types(ρT), instances

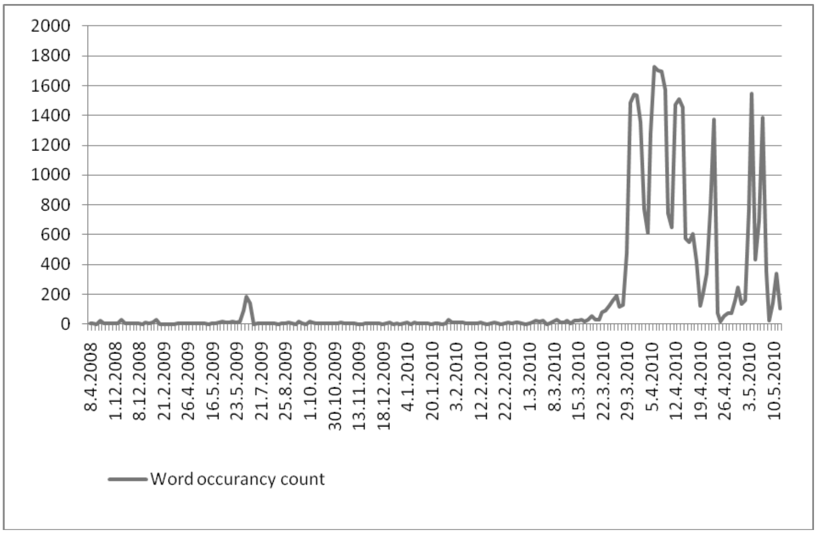


Fig. 3. Daily occurrence frequencies for the "iPad" term, showing the sought after spikes in word occurrence frequencies. The spikes determine that the word in question is not used regularly within the wider domain and that indeed it is the focus of reporting, therefore proving to be a domain entity.

to relations (ρR) and values of each attribute of each instance(ρA). Usually an ontology is thought of as a: set of concepts, relationships between concepts and a set of instances assigned to concepts. The formalization process needs to one of these tags to every word (chunk) in the processed sentence.

The first step in the formalization of the senteces is the creation of a semantic network that represents the relationships between individial words. Semantic networks is a term that encompasses a family of graph-based representations which share a common set of assumptions and concerns. A visual representation is that of a graph where node connections are the relations between concepts. Nodes and connections are labelled to provide the necessary information about the concept and type of association. The usual transformation from natural language to the semantic network is the mapping of nouns to concepts and verbs to the connections between them. The problem can be viewed in the following manner: a sentence without the punctuation marks is a sequence of words $(w_1 \dots w_n)$ and it is the task of the transfomation to a semantic network to assign each word to its appropriate type (concept or link). Therefore the transformation is essentially a classification task: types of semantic network building blocks are the classes and the classification method assigns each word (chunk) to its appropriate type. The required granularity of the classification is on the level of coarse grained classes. The transformation uses a word associated with its POS tag to map it to its type class. We will demonstrate this procedure an the example sentence:

The iPad has a 9.7-inch glass screen.

The w_p vector contains the words with their appropriate POS tags: $w_p = \{The: P, iPad: N, has: V, a: P, 9.7: Nr \ldots\}$, the Types (w_p) function maps the sentence to a semantic net representation shown on Fig. 4.

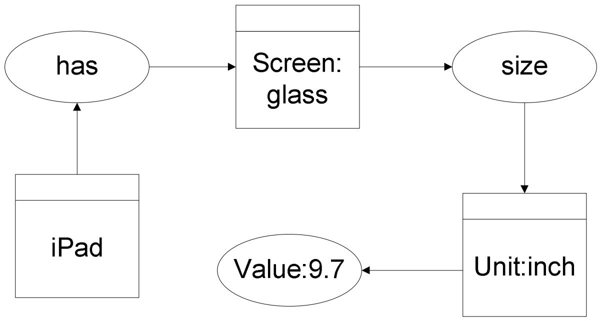


Fig. 4. The semantic network representation of the sentence "The iPad has a 9.7-inch glass screen"

We are using a supervised approach with a labeled training set for the classification function Types (w_p). The analysis of the news item has shown that the limited lenght of the syndicated news items is summarized and that the sentences are generally similar in structure. The approach selected is knowledge based. The knowledge that the function uses is essentially a machine readable dictionary of manually annotated sentences. For this purpose a manual selection of the most common sentence types in the gathered news items has been manually annotated. This is the training set for the naive Bayes classifier. A Naive Bayes classifier is a simple probabilistic classifier based on the Bayes's theorem [11]. It calculates the conditional probability of each type T_i of a word with the use of local word features f_j in the context. The type T is chosen as the most appropriate type from the formula:

$$T = \underset{T_i \in Types(w)}{\operatorname{argmax}} P(T_i | f_1, \ldots f_m) = \underset{T_i \in Types(w)}{\operatorname{argmax}} \frac{P(f_1, \ldots, f_m | T_i)\, P(T_i)}{P(f_1, \ldots, f_m)}$$
$$= \underset{T_i \in Types(w)}{\operatorname{argmax}} P(T_i) \prod_{j=1}^{m} P(f_j | T_i)\,. \qquad (2)$$

Parameter $m$is the number of features and the final formula is based on the assumption that the features are conditionally independent from the type. The probabilities $P(T_i)$ and $P(f_j|T_i)$ are estimated as the occurence frequencies in the training set of type T_i and feature f_j in the presence of the typeT_i.

The goal of the semantic network representation is the construction of the subject-predicate-object *triples* that are used for the semantic network and for the ontology learning process.

4 Ontology Learning

An ontology can be viewed as just another class of models which needs to be expressed in some kind of hypothesis language [1]. Ontology learning can be addressed via different tasks: learning the ontology concepts, learning the relationships between concepts or populating an existing ontology. Formally the ontology learning tasks are defined in terms of mapping between ontology components, where some of the components are given and some are missing and we want to induce the missing ones [1]. Typical ontology learning scenarios are: (1) inducing concepts, (2) inducing relations, (3) ontology population, (4) ontology generation and (5) ontology updating.

The approach we are presenting can be used for the ontology learning scenarios 2, 3 and 5. Relations inducing is presented on the example from section 3.4 where a new relationship of *has a glass screen* has been induced for the hardware device type tablet computer, the iPad.The main problem in the transformation from the representation, of the relationships between individual words in a sentence, in the form of a semantic net and that of the ontology is again a mapping one. The entities, in the semantic network, need to be mapped to either concepts or instances. For these purpose the named entity recognition module was included in the preprocessing workflow. The words tagged as named entities or domain terms as actually the instances of ontology subjects. However additional information is required for the mapping to the exact concept. In some cases the news items are manually tagged with a category tag, usually in the form of one or more words that begin with the number/hash sign and are positioned at the end of the news item category. For example: “Title about apples #fruit”, the #fruit tags this news item in the fruit category, therefore an apple is a type of fruit. However, when this information is missing or is too ambiguous or can’t be mapped to an ontology concept, additional information is required for the mapping. This can only be retrieved from the so called *definition* statement. A definition statement is a sentence that clearly defines a named entity to be something. Most common types of definition statements are the

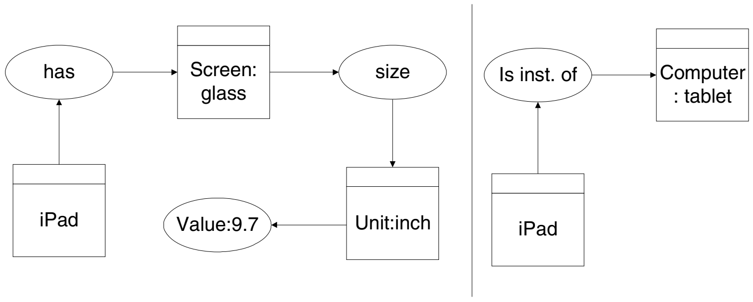


Fig. 5. The semantic network representation of two sentences: "The iPad has a 9.7-inch glass screen" and “iPad is a tablet computer.”These two networks can be combined as they provide the necessary information for the ontology population. The latter maps the entity “iPad” to the ontology concept of tablet computer and the former provides additional information about that instance (and the concept itself).

consequence (first the named entity then the definition) and the causal type (first the definition then the entity). Typical examples of the types respectively are the following two sentences: “Apples are a type of fruit!” and “A typical example of a garden fruit is the apple”.

Fig. 5 shows the two semantic representations from two news items sentences. These two provide enough information to be able to map the recognized entity (iPad) to the appropriate ontology concept. A computer is a domain concept and combined information of the two nets provides this information: iPad is an instance of a domain subject; it has a domain object of a screen with the properties of glass and size. This is all the necessary information that is required in order to enhance the ontology with a new instance and associate it with additional (previously unknown) information.

5 Conclusion

We have shown that a semi-automated approach for ontology construction/ population, based on natural language processing of concise passages is successful. The necessary prerequisite is foremost the amount of the text passages that are used as data sources. The names entity recognition, based on word use trend analysis, requires as long a history of news items as possible. This is essential in order to prevent false positives on entity recognition. However this is not something that involves a lot of human interaction. Once the crawler has been developed it is only a matter of time until the amount of news gathered is high enough to allow the accurate recognition of named entities. For the purposes presented here it is essential that entities are recognized correctly because they are the basis for the building of the semantic networks, that represent a formal knowledge representation of the sentences that contain or refer to named entities. A significant role in the entity recognition is played by the tokenization and POS tagging modules. However trivial the task of tokenization seems it is in fact a serious problem. In fact in some cases it is impossible to tokenize successfully if the context information is not evaluated as well. Therefore it was decided that the tokenization would be combined with the various modules of our ensemble tagger to provide the necessary context. The ensemble tagger plays an important role in the hybrid approach we have chosen for the POS tagging.

The semi-automated nature of the approach is expressed mostly in the construction of the initial ontology and the verification of the results. While the full examination of the successfulness of the approach has not been completed, the initial results show that with a sufficient level of development of the initial ontology, the approach provides very usable results. The merit is shown on the fact that the approach provides an automated means of population the ontology with new instances and their information. That means the ontology is constantly updated with real world developments and changes. The updating is a tedious task for knowledge engineers. If the knowledge represented in any formal means (ontology is a type of knowledge formalism) is not up to date then its usefulness is degraded linearly or even exponentially according to the age of the data.

For the future work we are planning an extensive evaluation of both the supervised and unsupervised uses of the presented approach. Additionally we will explore the possibilities of enhancing the approach in such a way that human supervision will be further reduced. The final goal is to reduce the human involvement in the ontology population process to its minimum.

References

1. Davies, J., Studer, R., Warren, P.: Semantic Web Technologies: Trends and Research in Ontology-based Systems. John Wiley & Sons Ltd., Great Britain (2006)
2. RSS 2.0 Specification, `http://www.rssboard.org/rss-specification`
3. Extensible Markup Language (XML) 1.0, `http://www.w3.org/TR/REC-xml/`
4. Heydon, A., Najork, M.: A scalable extensible web crawler. In: Proceedings of the Eight World Wide Web Conference, pp. 219–229 (1999)
5. Brewington, B.E., Cybenko, G.: How Dynamic is the Web. In: Proceedings of the Ninth International World Wide Web Conference, pp. 257–276 (2000)
6. Chakrabarti, S., van den Berg, M., Dom, B.: Focused crawling: a new approach to topic-specific Web resource discovery. In: Proceedings of the Eight International Conference on World Wide Web, pp. 1623–1640 (1999)
7. Grefenstette, G., Tapanainen, P.: What is a word, what is a sentence? Problems of tokenization. In: 3rd International Conference on Computer Lexicography, pp. 79–87 (1994)
8. Meir, R., Rätsch, G.: An introduction to boosting and leveraging. In: Mendelson, S., Smola, A.J. (eds.) Advanced Lectures on Machine Learning. LNCS (LNAI), vol. 2600, pp. 118–183. Springer, Heidelberg (2003)
9. Brants, T.: TnT – A Statistical Part-of-Speech Tagger. In: Proceedings of the Sixth Conference on Applied Natural Language Processing, pp. 224–231 (2000)
10. Ehrig, M., Haase, P., Hefke, M., Stojanovic, N.: Similarity for ontologies – A comprehensive framework. In: Proceedings of the 13th European Conference on Information Systems (2004)
11. Navigli, R.: Word Sense Disambiguation: A Survey. ACM Comput. Surv. 41(2), 1–69 (2009)

A Process to Derive Domain-Specific Patterns: Application to the Real Time Domain

Saoussen Rekhis[1], Nadia Bouassida[1], Claude Duvallet[2], Rafik Bouaziz[1], and Bruno Sadeg[2]

[1] MIRACL-ISIMS, Sfax University, BP 1088, 3018, Sfax, Tunisia.
`{saoussen.rekhis, raf.bouaziz}@fsegs.rnu.tn, nadia.bouassida@isimsf.rnu.tn`

[2] LITIS, UFR des Sciences et Techniques, BP 540, 76 058, Le Havre Cedex, France.
`{claude.duvallet, bruno.sadeg}@univ-lehavre.fr`

Abstract. Domain Specific Design Patterns are sets of objects and components that capture the experts knowledge and that can be used in a specific software domain. They provide for a higher software quality and a reduced development cost. However, their design remains a difficult task due to the generality and variability they must encompass, in order to be instantiated for various applications in the domain. This paper presents a design process that generates domain specific design patterns from a set of concrete application designs. The proposed process defines unification rules that apply a set of comparison criteria on various applications in the pattern domain. In addition, domain requirements, constraints and invariants are extracted and then confronted to the pattern, in order to validate it. The process integrates bottom-up and top-down approaches and assists the designer in the construction of domain specific patterns. Finally, the design process is illustrated and evaluated through the design of a pattern in the real-time domain.

Keywords: bottom-up process, top-down process, domain specific design pattern, real-time application.

1 Introduction

Design patterns [10] are recognized as a means for design reuse. They present a successful mechanism adopted by the object oriented community to capture and promote best practices in the software design. They promise several reuse benefits, such as high quality and a reduced development cost. These benefits have spread the adoption of patterns to many domains including human computer interaction, security, hypermedia and real-time systems.

However, due to the fact that general patterns are too abstract, their use can result in systems that do not correspond to reality. Moreover, their instantiation remains a difficult task since it is hard to determine in which context or in which part of the system the patterns can be used. These reasons motivated several works on domain-specific patterns ([15] [13] [6]). In fact, a domain-specific design pattern is a collection of interrelated objects that represent particular software domain knowledge, and thus supports vertical reuse. It offers a flexible architecture with clear boundaries, in terms of well-defined and highly encapsulated parts that are in alignment with the natural constraints of the domain [15].

Nowadays, domain-specific design patterns receive special attention from the many research communities dealing with reuse and domain knowledge representation. One of the reasons for this tendency is the increasing variability of software systems and the need to acquire expertise in the different evolving domains. Nevertheless, the difficulty of the domain specific patterns development and specification slows their expansion. This is due essentially to the fact they have to incorporate flexibility and variability in order to be instantiated for various applications in the domain. As a result, there is a need for a design process that guides the domain-specific patterns development and defines rules to facilitate their specification.

Several researchers have been interested in domain-specific patterns. However, most of them deal with patterns representation ([7][11][13][9]) and only few works are interested in patterns development processes ([17] [5]).

In this paper, we propose a new domain-specific pattern development process based on a UML profile appropriate to pattern specification and reuse [19]. This process has to guide the developer in determining the different aspects in which applications may differ and the common aspects, which remain unchanged in all these applications. The proposed process is illustrated through the development of a pattern in the real-time domain: the sensor pattern.

This paper is organized as follows. Section 2 describes related works. Section 3 presents the domain-specific pattern process based on our proposed language for pattern representation. Section 4 applies our development process to derive a pattern for the real-time domain. Firstly, it presents three applications (an air traffic control system [12], a freeway traffic management system [1], and an industrial regulation system [18]). Secondly, it applies the unification phases to construct the domain-specific pattern i.e., sensor pattern. Finally, section 5 concludes the paper and outlines future works.

2 Related Work

Domain specific patterns are similar to object-oriented frameworks, since they offer design reuse for a particular domain. However, they differ in their granularity since frameworks reuse all the domain architecture while domain specific patterns offer only small structures reusable in a certain domain. Thus, to propose a domain specific patterns development process, we have been inspired in our work from frameworks development processes, which we briefly present in Subsection 2.2. The domain specific pattern development processes are overviewed in Subsection 2.1.

2.1 Patterns Development Processes

Domain-specific patterns development processes can be classified as either bottom-up or top-down.

A bottom-up process starts form a set of applications representing the domain and identifies common and variable elements. The purpose of examining sample applications is to create a generic pattern that is understandable and easy to reuse. Note that the bottom-up design process works well when a domain is already well understood, for example, after some initial evolutionary cycles [3]. However, there is no guarantee that all domain requirements are met.

A top-down process starts from a domain analysis and then constructs the design pattern. This means that the design is driven starting from functional requirements towards solution alternatives. Top-down development processes represents the best solution when the domain has not yet been sufficiently explored [3]. However, this type of processes discourages many pattern developers since it is time-consuming and it lacks strategies for the domain requirements analysis.

Raminhos [17] and Caeiro [5] describe steps to follow when a developer wants to builde a domain-specific pattern. Raminhos [17] focuses on the definition of a systematic process to guide developers to fill an analysis pattern template. This template is proposed as a combination of the common features of existing templates. This process is depicted as an UML activity diagram where each activity helps filling in one entry of the template. Caeiro [5] focuses on e-learning domain and decomposes the top-down patterns design process on two steps. According to this process, the designer starts with the determination of e-learning basic activities. In the next step, it is necessary to define the control flow between the different activities. For each activity, it is necessary to describe the actors involved, the environment and the way in which the actors are going to interact.

In fact, these processes do not provide an efficient assistance for patterns development. In addition, they do not clearly identify nor they guide the developer in finding the pattern fundamental and variable elements. We believe that it is necessary to define a process that allows to distinguish between pattern fixed and variable parts, similarly to processes proposed for bottom-up framework development [20][8] [3].

2.2 Frameworks Development Processes

Frameworks represent other forms of reusable assets for vertical reuse. They represent an extension to object-oriented designs, that help reuse at a level of abstraction higher than classes and patterns. In fact, an object-oriented framework represents a software architecture that captures several applications' behaviours in a specific domain. It is composed of a set of concrete and abstract classes with their relations. In general, several patterns may be used to structure and to document a particular framework. Therefore, the unification rules [8] [3] defined to generate a framework class diagram can be adapted to create a domain-specific pattern.

Fontoura et al. [8] propose a development process by considering a set of applications as viewpoints of the domain, i.e., each application represents a different perspective of the framework domain. The process defines a set of informal unification rules that describes how the viewpoints can be combined to compose a framework. However, the resulting framework consists of an OMT class diagram and does not specify the interactions between its objects. Furthermore, this process assumes that all the semantic inconsistencies between the viewpoints have been solved. Thus, it does not treat semantic problems (e.g. equivalent terms, variable terms, and so on).

The bottom-up development framework process, proposed by [3], generates a framework design by defining formal unification rules that identify automatically the commonalities and differences between a set of application designs based on UML. The process is composed of three main steps. The first step extracts the domain specifications and potential uses of the framework through unification of the use case diagrams of different applications in the domain. The second step models the framework static features through unification of the class diagrams of the given applications. The last step extracts the dynamic framework features through the unification of sequence diagrams. The above three unification steps use a set of unification rules based on semantic comparison criteria. Nevertheless, it is impossible to be sure that the resulting framework fulfils all domain constraints and requirements; this depends on the application designs used from the beginning of the unification process.

In the following section, we propose a domain-specific pattern design process that combines the bottom-up and top-down approaches in order to facilitate the pattern developers' work and to specify patterns with a better quality. In fact, the process uses three primary steps: (i) the analysis of domain functionalities and requirements, (ii) the generation of patterns from the unification of a set of applications and (iii) the validation of the resulting patterns.

3 The Development Process for Domain-Specific Patterns

The development process generates a domain-specific pattern represented with the UML profile we have proposed[19]. The objective of this section is two-fold. Firstly, it describes the stereotypes defined in the proposed UML profile [19] that guides a designer in instantiating a pattern to derive a specific application. Secondly, it proposes a process that guides the development of domain-specific patterns. This process adopts, a bottom- up approach on one hand; since it generates a pattern from a given set of applications and it completes the approach with a domain requirements analysis that aims to validate the pattern, on the other hand.

3.1 UML Profile for Domain-Specific Design Patterns

The proposed profile extends UML 2.2 [14] with new concepts related to domain specific design patterns. The main motivations behind these extensions are to facilitate the comprehension of design patterns and to distinguish the pattern's fundamental elements from the optional and variable elements which change from an application to another.

Table 1 describes the extensions that we propose to take into account these new concepts.

Table 1. UML extensions suitable for domain-specific patterns representation

≪**optional**≫ applied to the Feature UML Metaclass.	This stereotype is used to specify optional features in UML class diagram. When an attribute (or a method) is stereotyped ≪**optional**≫, then it can be omitted in a pattern instance. Each method or attribute which is not stereotyped ≪**optional**≫ in a fundamental class means implicitly that it is an essential element, i.e. it plays an important role in the pattern.
≪**mandatory**≫ applied to the UML Metaclasses: Class, Association, Interface, Lifeline, ClassAssociation.	This stereotype is used to specify a fundamental element (association, aggregation,...) that must be instantiated by the designer when he models a specific application. A fundamental element in the pattern is drawn with a highlighted line like this class ▤. Besides, each pattern element which is not highlighted means that it is an optional one, except the generalization relation that permits to represent alternative elements. All the attributes and methods of an optional class are implicitly optional.
≪**extensible**≫ applied to the UML Metaclasses: Class, Interface and ClassAssociation.	This stereotype is inspired from extensible tagged value proposed in [4]. It indicates that the class interface may be extended by adding new attributes and/or methods. Moreover, we propose to define two properties for the extensible stereotype specifying the type of element (attribute or method) that may be added by the designer. - extensibleAttribute tag: It takes the value false, to indicate that the designer cannot add new attributes when he instantiates the pattern. Otherwise, this tag takes the value true. - extensibleMethod tag: It indicates that the designer may add new methods when instantiating the pattern. The default value is true.
≪**variable**≫ applied to the UML Metaclass: Method.	This stereotype is inspired from variable tagged value proposed in [4]. It indicates that the implementation of the method varies according to the pattern instantiation.

3.2 Pattern Development Process Description

The process presented in Figure 1 as an activity diagram, illustrates a set of steps which help the pattern designer to construct the domain-specific design pattern. This process is decomposed into five essential phases. The first phase, which belongs to the domain analysis, identifies the most important functionalities of the domain, i.e. domain sub-problems. The second step collects the requirements, invariants and constraints that have to be fulfilled by the different functionalities. The third phase goes through the decomposition of the different applications according to the already identified functionalities. The fourth step consists in a unification of a set of application designs in order to capture the information involved in the domain. The last step checks if all the requirements and constraints are fulfilled by the pattern, in order to validate it. This step is crucial since the pattern will be reused in developing further applications of the same domain. A brief description of these phases is given below:

– **Identification of domain functionalities :**
  After finding out the boundaries of the domain and providing examples of the applications, the first step is to decompose the domain into several sub-problems in order to decrease its complexity. Each sub-problem has one main functional goal to achieve the domain *functionality* (e.g., data acquisition and data control are functionalities of the real-time domain). The most important functionalities related to a domain are identified through the collection of information from experts and stakeholders. The different viewpoints of experts and stakeholders must be compared in order to consider only the similar needs corresponding to the domain functionalities.
– **Identification of requirements :**
  This step is performed by the patterns designer. It requires many efforts since it implies a top-down process allowing the refinement of the functionalities belonging to the domain. In fact, each previously identified functionality is decomposed iteratively into functions until reaching a level at which the functions become elementary. This refinement allows the identification of domain concepts related to each functionality as well as the associated domain constraints and invariants. For example, the *sensor* and *measure* represent two concepts belonging to the real-time domain.
– **Decomposition of applications :**
  The already identified domain concepts help the designer to decompose the applications examples. That is, the designer has to find the relations between the classes of the application and the domain concepts related to each functionality, in order to determine the application fragments. In fact, a good decomposition results in application fragments having significant purposes.
– **Fragments unification of applications :**
  In order to unify the different application fragments and to derive the domain specific pattern, a set of rules are used. In fact, the unification rules are an adaptation of those proposed by Ben Abdallah et al. [3] for frameworks development. Thus, we propose to fuse the common classes to all the applications and to derive the fundamental elements of the pattern. We add then the appropriate classes to the applications as variable elements.

  The unification rules use semantic correspondence criteria to compare class names, attributes and operations.

  The class name comparison criteria express the linguistic relationships among class names and consist in the following three relations:

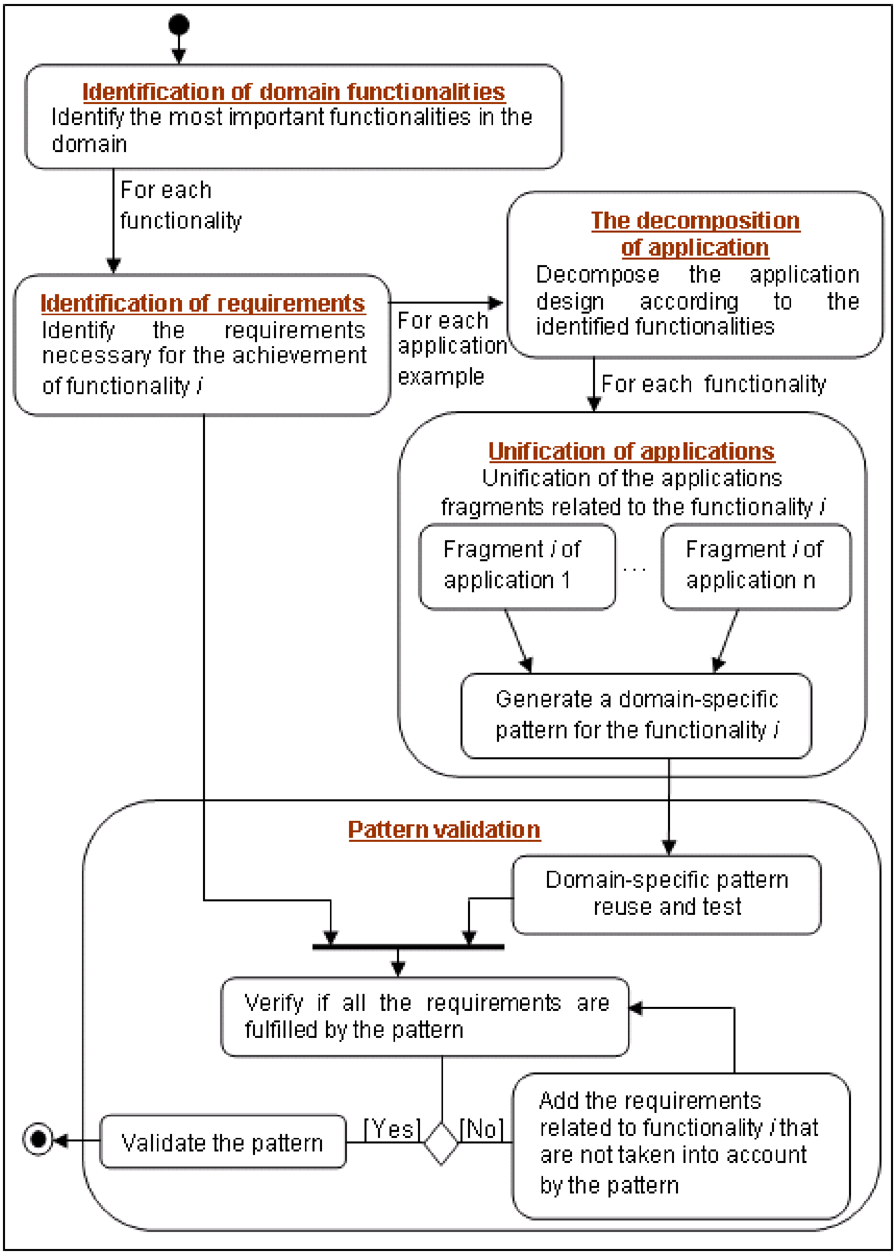

Identification of domain functionalities
Identify the most important functionalities in the domain
For each functionality
Identification of requirements
Identify the requirements necessary for the achievement of functionality *i*
For each application example
The decomposition of application
Decompose the application design according to the identified functionalities
For each functionality
Unification of applications
Unification of the applications fragments related to the functionality *i*
Fragment *i* of application 1
...
Fragment *i* of application n
Generate a domain-specific pattern for the functionality *i*
Pattern validation
Domain-specific pattern reuse and test
Verify if all the requirements are fulfilled by the pattern
[Yes]
[No]
Validate the pattern
Add the requirements related to functionality *i* that are not taken into account by the pattern

Fig. 1. The development process for the domain-specific patterns

- N_equiv(CA_{1j},...,CA_{nj}) means that the names of the classes are either identical or Synonym.
  Note that the class C in the fragment_j of application A_i is represented by CA_{ij} and the fragment_j is the part of application model relative to functionality_j.
- N_var(CA_{1j},...,CA_{nj}) means that the names of the classes are a variation of a concept, e.g. mobile-sensor, passive-sensor, active-sensor.
- N_dist(CA_{1j},...,CA_{nj}) means that none of the above relations holds.

The attributes comparison criteria use the following three relationships to compare the attribute names and types:

- Att_equiv(CA_{1j},..., CA_{nj}) means that the classes have either identical or synonym attribute names with the same types.
- Att_int(CA_{1j},..., CA_{nj}) means that the classes CA_{1j},..., CA_{nj} have common attributes.
- Att_dist(CA_{1j},..., CA_{nj}) means that none of the above relations holds.

The operation comparison criteria use three relations (Op_equiv(CA_{1j},...,CA_{nj}), Op_int(CA_{1j},...,CA_{nj}), Op_dist(CA_{1j},...,CA_{nj})) to compare the operation names and signatures (returned types and parameter types). These relations are defined in a way similar to the attribute comparison relations.

The design of a domain-specific pattern class diagram is guided by the following unification rules:

R1. If a class is present in all the applications with equivalent attributes i.e. Att_equiv(CA_{1j},..., CA_{nj}) and methods i.e. Op_equiv(CA_{1j},...,CA_{nj}), then it is added to the pattern as a fundamental class with a highlighted border.

R2. If a class is present in several applications, then it is added to the pattern as an optional class if the domain coverage ratio (R_{dc}) is superior or equal to 2/3 [3]:

$$R_{dc}(C) = \frac{Number\ of\ occurrences\ of\ C\ in\ applications\ fragments}{Number\ of\ applications}$$

Otherwise, if the class is present in few applications (i.e. $Rdc < 2/3$), then it is too application-specific. Thus, if it is added to the pattern, it may complicate unnecessarily the pattern comprehension. Then this class will be added by the designer in a pattern instantiation.

R3. If a set of classes belonging to all the applications share equivalent attributes and/or methods i.e. Att_int(CA_{1j},..., CA_{nj}) and/or Op_int(CA_{1j},...,CA_{nj}), then there are four cases:

Case 1: if N_var(CA_{1j},...,CA_{nj}), then an abstract class that have common attributes and methods is added to the pattern as a fundamental class. Moreover, a set of sub-classes inheriting from the abstract class and containing the attributes and methods that are specific to only one application are added to the pattern.

Case 2: if N_equiv(CA_{1j},...,CA_{nj}), then a fundamental class is added to the pattern with common attributes and methods considered as fundamental elements, and the distinct attributes and/or methods generated as optional elements if they are pertinent for the pattern domain.

Case 3: if N_dist(CA_{1j},...,CA_{nj}), then a fundamental class is added to the pattern if the classes (CA_{1j},...,CA_{nj}) play the same role related to one domain concept considered in the requirements identification step. Otherwise, these classes are too application specific and they are not relevant to the pattern.

Case 4: if it exists distinct attributes and/or methods that are not pertinent for the pattern domain, then a fundamental class is added to the pattern with minimal number of attributes and/or methods and it is stereotyped ≪**extensible**≫ in order to indicate that the class can be extended when reusing the pattern.

R4. If a method exists in all the applications with the same name but with different signatures, then it will have a corresponding method in the pattern with an undefined signature and it is stereotyped ≪**variable**≫.

R5. If two or more classes are transferred in the pattern, then all their relations (aggregation, inheritance, association) will be maintained in the pattern.

Note that the resulting domain-specific design pattern is based on our UML profile appropriate to domain pattern specification [19].

– **Pattern validation:**
After the applications unification phase, the resulting domain-specific pattern is validated in two steps. First, the obtained domain-specific pattern is instantiated and confronted with the original application fragments. Second, the completeness of the pattern is verified by checking if all domain functionality requirements and constraints are fulfilled by the obtained pattern. In fact, the domain-specific pattern development process is iterative and it will not stop until all the requirements are taken into account by the pattern.

4 A Case Study: The Real-Time Sensor Pattern Development

To evaluate the proposed process and to illustrate the unification rules of the class diagram, we propose to develop a pattern appropriate for the real-time (RT) domain. The most important motivations of this choice is to reduce the complexity of RT applications design. In fact, RT applications must be able to meet RT constraints, i.e. they have to guarantee that each action (transaction) meets its deadline, and that data are used during their validity interval. Thus, it is necessary to give a great importance to RT applications design.

Note that all RT applications share a common behaviour: they monitor and control the values acquired from the environment through a set of components that work together to achieve a common objective. By examining these applications, we distinguish three principal RT functionalities: the RT data acquisition through sensors, their control and their RT use. We focus in this paper on modelling the acquisition functionality through the definition of the RT sensor pattern. The primitive requirements of this functionality are identified through a refinement process. Thereby, the RT data acquisition sub-problem is decomposed into three elementary functions: (i) the sensor observes the environment;

(ii) it detects a physical phenomenon (temperature, pressure, speed, etc.) and (iii) transmits the acquired measure periodically to the computer unit. A set of domain concepts such as sensor, measure, and so on are identified during the refinement step. These concepts facilitate the applications decomposition. That is, the RT applications fragments related to data acquisition functionality are defined through the identification of the classes that play the roles of Sensor, Measure and Observed Element as well as the associated entities.

In the applications fragments unification, we began by choosing three RT applications: an air traffic control system [12], the COMPASS system [1], which is a freeway traffic management system, and an industrial regulation system, which allows the control of the water levels in tanks [18]. Afterward, we construct a dictionary that defines the semantic relations (equivalence, variation,...) between nouns of classes, attributes and methods. The dictionary is used by the unification rules to obtain the RT sensor pattern.

4.1 Description of RT Applications

In this section, we describe, briefly, the three RT applications examples:

4.1.1 The COMPASS System: The freeway traffic management systems have become an important task intended to improve safety and provide a better level of service to motorists. In the following, we describe an example of a freeway traffic management system: COMPASS [1]. We focus precisely on modeling the compass data acquisition subsystem as illustrated in figure 2.

The current traffic state is obtained from the essential sources: inductance loop detectors and supervision cameras. In fact, vehicle detector stations use inductance loops to measure speeds and lengths of vehicles, traffic density,i.e. number of vehicles in a road segment and occupancy information. These processed data are then transmitted periodically to the Central Computer System. Whereas the supervision cameras are used to confirm the reception od data through the vehicle detector stations and to provide information on local conditions which affect the traffic flow. The computer system uses the acquired data stored in a real-time database to monitor traffic and to identify traffic incidents.

For each measure taken from the environment of this system and stored in the database, the designer must specify the value, the timestamp and the validity interval to verify the temporal consistency of the collected traffic data. For example, the value of the vehicle speed measure is temporally consistent as long as it is no more than twenty seconds. In addition, the designer must specify the minimum and maximum thresholds of each taken measure in order to determine the abnormal values for which COMPASS system may detect an incident.

As illustrated in Figure 2, vehicle speed, vehicle length, traffic volume and occupancy constitute instances of the Measure class. Moreover, the Vehicle and the RoadSegment classes represent the physical elements that are supervised by the passive sensors. The speed and length measures are relative to the Vehicle

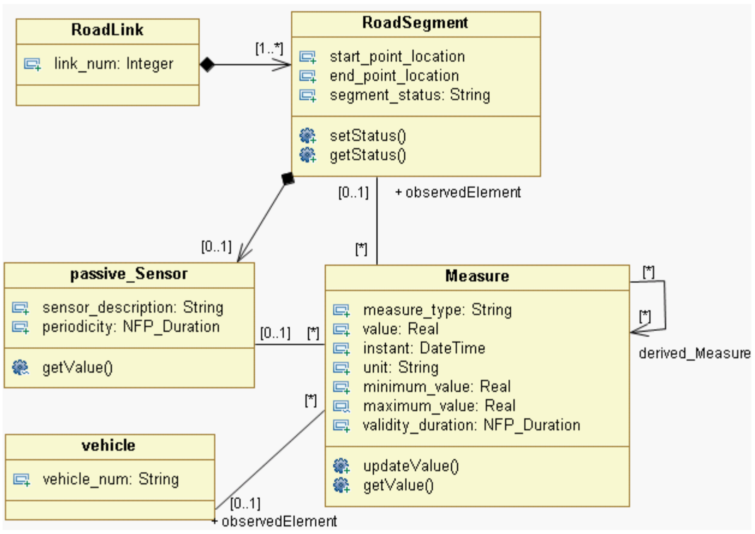


Fig. 2. The data acquisition design of a freeway traffic management system

class. Whereas the traffic density and occupancy measures are relative to the RoadSegment class.

4.1.2 Air Traffic Control System: The aim of the air traffic control is to avoid collisions and to organize the flow of traffic. It consists of a large collection of data describing the aircrafts, their flight plans, and data that reflect the current state of the controlled environment [12]. This includes flight information, such as aircraft identification, speed, altitude, origin, destination, route and clearances. In fact, each aircraft has three basic measures which are speed, altitude and location and one derived measure which is aircraft path. The basic measures are periodically updated to reflect the state of an Aircraft. The derived measure is calculated based on altitude and location values, in order to verify if the aircraft deviates from a predetermined path. Note that, altitude, speed, location and path constitute instances of Measure class. All these measures values are published periodically by sensors supervising the aircrafts controlled elements.

We present in figure 3 a fragment of an air traffic control application relative to the data acquisition functionality.

4.1.3 Industrial Control System: The purpose of the water level control of an industrial regulation system is to monitor and to control the water levels in tanks, ensuring that the actual water level of tank i is always between Low-level and High-level. [18]. If some of the tanks do not satisfy their boundary constraints, then the system tries to resolve the problem internally, for example,

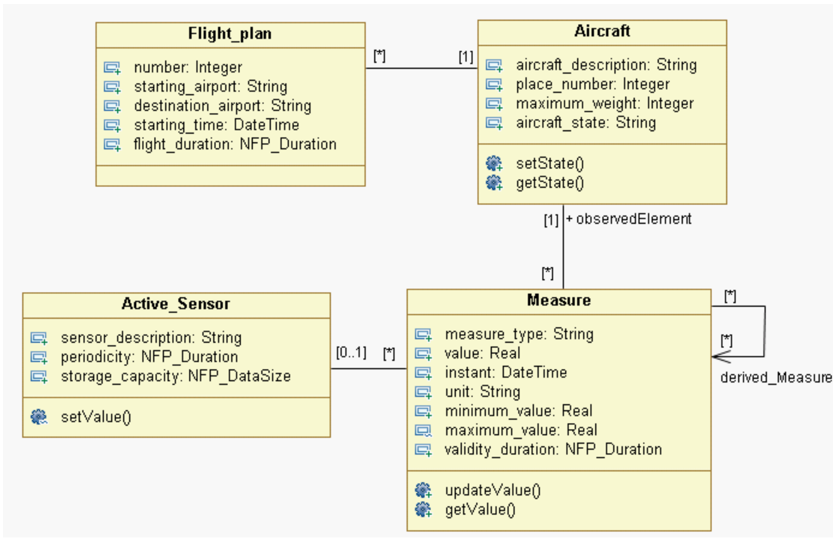


Fig. 3. The data acquisition design of an air traffic control system

by rebooting the system. However, if the problem cannot be resolved internally, then the system requires a special treatment of an external exception handler.

The actual levels of the different tanks are measured periodically by boundary sticks sensors. Each acquired measure is characterized by a timestamp, a validity duration that represents the time interval during which the measure is considered valid and a minimum value and a maximum value of the water level in the tank. When the water height in the tank reaches its low (or high) desirable limit, then the filling (or emptying) faucet is activated to inject water into the tank (or to drain water from the tank).

Figure 4 illustrates the data acquisition functionality of the water level control system. It indicates that this application controls one type of elements (tanks) and monitors one type of measure, which is the water height in tanks, through the boundary sticks passive sensors.

4.2 RT Sensor Pattern Construction

The RT sensor pattern class diagram is shown in Figure 5. It is obtained by applying the unification rules as indicated in the following:

- N_var (Measure A1, Measure A2, Water_height A3), Att_equiv (Measure A1, Measure A2, Water_height A3) and Op_equiv (Measure A1, Measure A2, Water_height A3). The rule R2 is applied in this case and a fundamental class Measure is added to the sensor pattern.
- N_var (InductanceLoop_sensorA1, Active_sensorA2, BoundaryStick_sensorA3), Att_int (InductanceLoop_sensorA1, Active_sensorA2, BoundaryStick_sensorA3) and Op_dist (InductanceLoop sensorA1, Active sensorA2, BoundaryStick sensorA3). The rule R3 is applied and a set of inheriting classes from the

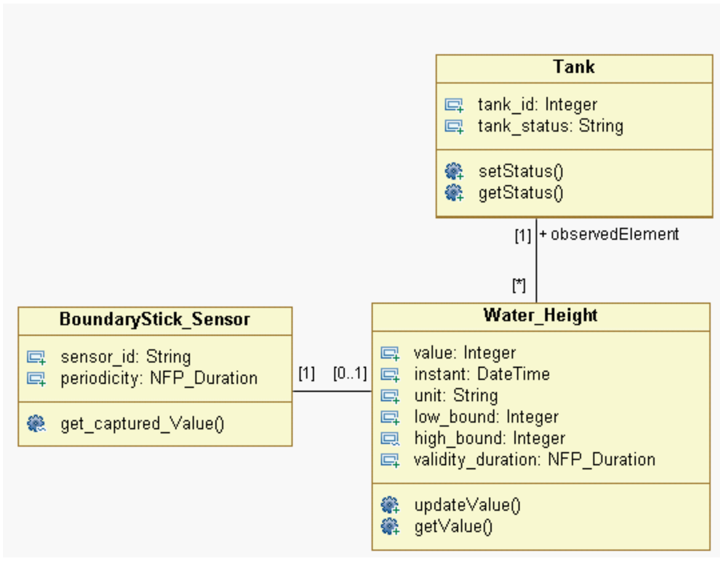


Fig. 4. The data acquisition design of a water level control system

abstract Sensor class are added to the pattern. The Sensor class has the common attributes and it is stereotyped ≪**extensible**≫. The methods (setValue and getValue) belong respectively to the sub-classes Active_Sensor and passive_sensor.

– N_dist (RoadSegmentA1, VehicleA1, AircraftA2, TankA3) and Att_int (RoadSegmentA1, VehicleA1, AircraftA2, TankA3). The rule R3 (cf. case 3) is applied and a fundamental class is added to the pattern because the classes (RoadSegmentA1, VehicleA1, AircraftA2, TankA3) have the same role which is observedElement in all the applications examples. The name of the fundamental class is relative to the role played by the corresponding applications classes.
– The rule R5 is applied to transfer the relations between classes in the pattern.

The obtained sensor pattern is valid since all the applications fragments are found after a pattern instantiation. Moreover, additional constraints and requirements, obtained from the requirements identification phase are added to the pattern. In fact, RT applications must use fresh data (timestamp + validity_duration of a measure $\leq$ current_time). Since the measure update depends on the use of sensor, the validity duration must exceed the periodicity of sensor acquisition data [16]. An OCL constraint is associated to the Measure class to define this condition.

In addition to the need of fresh data, RT applications have to use precise data in order to reflect the continuous change of the external environment. However, it seems difficult for the transactions to both meet their deadlines and to keep the database consistent. For this reason, the concept of data quality is introduced in [2] to indicate that data stored in RT databases may have some deviation from their values in the real world. Thereby, we propose to add the Maximum Data Error (MDE) attribute to the Measure class. This attribute represents a non functional property which specifies the upper bound of the error. It allows the system to handle the unpredictable workload of the database since an update transaction Tj is discarded if the deviation between the current data value and the updated value by Tj is less or equal than MDE. This attribute is of the same type as the value attribute.

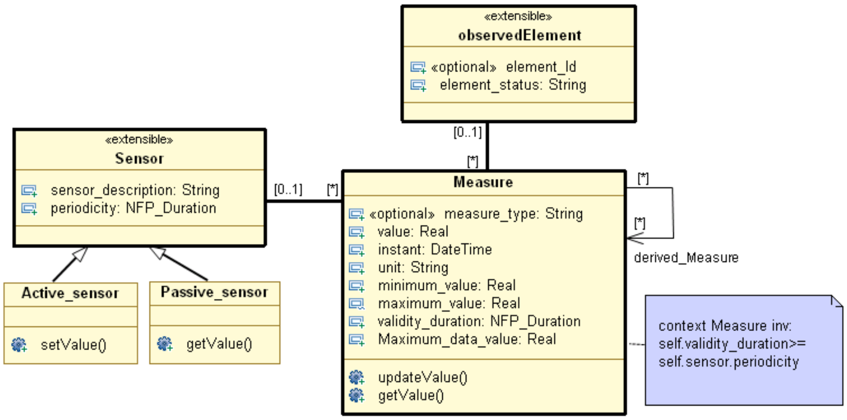


Fig. 5. Real time sensor pattern class diagram

5 Conclusion

The definition of reusable assets, such as patterns, components and frameworks, is usually performed by studying examples of existing systems and models for the domain. This approach, which is purely bottom-up, gives a view of the domain that is limited to what has been developed in the past. In order to meet this lack, the research areas of architectures and domain analysis advocate a top-down approach to achieve systematic reuse. Practice shows that a combination of both these approaches is often desired or needed. For this reason, this paper proposed an integrated pattern definition process that supports both the top-down and bottom-up approaches. It is distinguished from existing methods in two ways: 1) it is based on UML-profile language that visually distinguishes between the fundamental and variable elements of a pattern, and 2) it defines the different steps that must be taken to obtain a domain-specific design pattern as well as a precise set of unification rules that identifies the commonalities and differences between applications in the domain. The process was illustrated through the design of a sensor pattern for the real-time domain.

Our future work includes the definition of a process for pattern reuse and the development of a tool supporting a graphical representation of domain-specific design patterns and validating the application models against the relevant patterns models.

References

1. COMPASS website,
`http://www.mto.gov.on.ca/english/traveller/compass/main.htm`
2. Amirijoo, M., Hansson, J., Son, S.H.: Specification and Management of QoS in Real-Time Databases Supporting Imprecise Computations. IEEE Transaction Knowledge and Data Engineering 55(3), 304–319 (2006)
3. Ben-Abdallah, H., Bouassida, N., Gargouri, F., Hamadou, A.B.: A uml based framework design method. Journal of Object Technology 3(8), 97–120 (2004)
4. Bouassida, N., Ben-Abdallah, H.: Extending UML to guide design pattern reuse. In: AICCSA, pp. 1131–1138. IEEE, Los Alamitos (2006)
5. Caeiro, M., Llamas, M.,Anido, L.: E- learning patterns: an approach to facilitate the design of e-learning materials. In: 7th IberoAmerican Congress on Computers in Education (2004)
6. Couturier, V.: Patterns d'analyse pour l'ingénierie des systèmes d'information coopératifs. L'OBJET 11(4), 141–175 (2005)
7. Díaz, P., Aedo, I., Rosson, M.B.: Visual representation of web design patterns for end-users. In: AVI, pp. 408–411 (2008)
8. Fontoura, M., Crespo, S., de Lucena, C.J.P., Alencar, P.S.C., Cowan, D.D.: Using viewpoints to derive object-oriented frameworks: a case study in the web-based education domain. Journal of Systems and Software 54(3), 239–257 (2000)
9. France, R.B., Kim, D.K., Ghosh, S., Song, E.: A UML-Based pattern specification technique. IEEE Trans. Softw. Eng. 30(3), 193–206 (2004)
10. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, Reading (1995)
11. Kim, D.: The role-based meta-modeling language for specifying design patterns, pp. 183–205. Idea Group Inc., USA (2007)
12. Locke, D.: Applications and system characteristics. In: Real-Time Database Systems: Architecture and Techniques, pp. 17–26. Kluwer Academic Publishers, Dordrecht (2001)
13. Montero, S., Díaz, P., Aedo, I.: A semantic representation for domain-specific patterns. In: Wiil, U.K. (ed.) MIS 2004. LNCS, vol. 3511, pp. 129–140. Springer, Heidelberg (2005)
14. OMG: Unified Modeling Language: Superstructure Version 2.2, formal 2009-02-02 (February 2009)
15. Port, D.: Derivation of domain specific design patterns. In: USC-CSE Annual Research Review and Technology Week Presentations and Binder Materials (1998)
16. Ramamritham, K., Son, S.H., DiPippo, L.C.: Real-Time Databases and Data Services. Real-Time Systems 28(2-3), 179–215 (2004)
17. Raminhos, R., Pantoquilho, M., Araújo, J., Moreira, A.: A systematic analysis patterns specification. In: Manolopoulos, Y., Filipe, J., Constantopoulos, P., Cordeiro, J. (eds.) ICEIS, vol. (3), pp. 453–456 (2006)
18. Reinhartz-Berger, I., Sturm, A.: Utilizing domain models for application design and validation. Information & Software Technology 51(8), 1275–1289 (2009)
19. Rekhis, S., Bouassida, N., Bouaziz, R., Sadeg, B.: A UML profile for domain specific patterns: application to real-time. In: DE@CAISE 2010: the Domain Engineering Workshop of the 22nd International Conference on Advanced Information Systems Engineering (2010)
20. Schmid, H.A.: Systematic framework design by generalization. ACM Commun. 40(10), 48–51 (1997)

A Sample Advisor for Approximate Query Processing

Philipp Rösch[1] and Wolfgang Lehner[2]

[1] SAP Research Center Dresden, Germany
philipp.roesch@sap.com
[2] Database Technology Group, Technische Universität Dresden, Germany
wolfgang.lehner@tu-dresden.de

Abstract. The rapid growth of current data warehouse systems makes random sampling a crucial component of modern data management systems. Although there is a large body of work on database sampling, the problem of automatic sample selection remained (almost) unaddressed. In this paper, we tackle the problem with a sample advisor. We propose a cost model to evaluate a sample for a given query. Based on this, our sample advisor determines the optimal set of samples for a given set of queries specified by an expert. We further propose an extension to utilize recorded workload information. In this case, the sample advisor takes the set of queries and a given memory bound into account for the computation of a sample advice. Additionally, we consider the merge of samples in case of overlapping sample advice and present both an exact and a heuristic solution. Within our evaluation, we analyze the properties of the cost model and compare the proposed algorithms. We further demonstrate the effectiveness and the efficiency of the heuristic solutions with a variety of experiments.

1 Introduction

Recent studies have revealed a rapid growth in current data warehouse databases regarding both the size and the number of queries. Additionally, more and more queries are of explorative nature where users browse through the data and search for interesting regions. [1]

In order to make the data exploration reasonably applicable, short response times are essential. However, the observed rapid growth conflicts with the need for short response times. A common solution for this problem is the use random samples. Initially, samples were mostly used for query optimization. In the last 10 years, however, the focus of database sampling has shifted more and more towards approximate query processing. Especially, precomputed and materialized samples provide large potentials due to their efficiency and their wide range of application. Several sampling schemes have been proposed that are optimized for different query types, like aggregation [2,3], group-by [4,5] or foreign-key joins [6,7]. While those sampling schemes provide great solutions for single (groups of) queries, the more general problem of automatic sample selection for a specific mixture of queries or even an entire workload of a database is still an open issue.

In this paper, we address the problem of finding a set of samples for precomputation and materialization. We focus on simple random samples as those samples are easy to use and to maintain. Moreover, they can be used for a broad range of queries. Our solution is a sample advisor that suggests a set of samples for a set of queries specified by an expert (referred to as *expertise-based sample configuration*). The sample advisor is based on a novel cost model to evaluate a sample for a given query. This cost model allows us to give advice on a sample for an individual query. We further propose an extension to utilize recorded workload information (referred to as *workload-based sample configuration*). In this scenario, the sample advisor selects from all the pieces of sample advices those that minimize the runtime of the workload and fit into a given memory bound. A second strategy of this workload-based sample advisor additionally considers the merge of samples in case of overlapping sample advice.

In summary, we make the following contributions:

- We propose a cost model for the evaluation of a sample for a given query. With this cost model, we can give a piece of sample advice for an individual query (Section 2.1).
- Based on the cost model, we show how to compute an expertise-based sample configuration (Section 2.2).
- We further propose two strategies (with and without merging sample advice) to find a good workload-based sample configuration. For both strategies, we present an exact and a heuristic solution (Section 3).
- With a variety of experiments, we analyze the properties of the cost model and compare the proposed algorithms. We further demonstrate the effectiveness and the efficiency of the heuristic solutions (Section 4).

In Section 5, we discuss related work, and finally, we summarize the paper in Section 6.

2 Sample Advisor

Our major focus is the recommendation of an expertise-based sample configuration. We now define what a sample for an individual query should look like. In more detail, since we focus on simple random samples, we show how to find a proper sample size for a given query.

2.1 A Cost Model for Sample Selection

The sample selection problem is related to the physical design problem for indexes or materialized views. However, in the case of samples, we have to face up to a new dimension: As the result of a sample-based query is only an approximation, we additionally have to take a certain error into account. This error can be an incompleteness—like for missing groups—or an inexactness—like for estimates of aggregates—and directly depends on the size of the sample.

Now, for the cost model, we have to discuss what makes up a good sample. Obviously, with a sample we want to achieve large decreases in the response times, and the memory cost should be low. These two goals ask for small sample sizes. At the same time, the estimates should be close to the actual values, and the results should be preferably complete. Clearly, these goals ask for large sample sizes. Hence, there is a conflict between the goals which has to be reflected by the cost model.

As a basis, we take the cost function of the DB2 Design Advisor [8] (bold part) and extend it by the approximation-related parts identified in this paper (italic part). The resulting cost model is:

$$\text{weight} = \frac{\textbf{decrease in response time} \cdot \textit{completeness}}{\textbf{memory cost} \cdot \textit{estimation error}} \,. \tag{1}$$

As can be seen, we append *completeness* (that we want to have) to the numerator and *estimation error* (that we don't like to have) to the denominator.

We now analyze the individual properties in more detail. Let N be the cardinality of the base data R, with $R = \{t_1, t_2, \ldots, t_N\}$. Further, let n be the cardinality of the sample S. Now, the sampling fraction f can be expressed as $f = n/N$. Moreover, let L denote the length of a tuple in the base data, while l is the length of a tuple in the sample.

Decrease in Response Time. For estimating the decrease in the response time, we make use of the simplified assumption that both the exact and the approximate query use table scans. Indeed, this assumption often holds in practice for the complex queries focused on in this paper. With this assumption the decrease in the response time Δt is proportional to $N - n$, and the relative decrease $\Delta t_{rel}(n)$ can be expressed as:

$$\Delta t_{rel}(n) = 1 - \frac{n}{N} = 1 - f \,. \tag{2}$$

Note that this function is independent from the dataset. It linearly decreases with increasing sample size.

Completeness. Let G be the set of groups defined by the query of interest. Then, $g_i \in G, i = 1...|G|$, denotes an individual group and $|g_i|$ denotes its size. Now, the probability p that at least one tuple of a group g_i is included into a sample of size n is given by:

$$p(g_i, n) = 1 - \frac{(N - |g_i|)!}{(N - |g_i| - n)!} \frac{(N - n)!}{N!} \,. \tag{3}$$

With this probability, the expected number of groups in the sample is

$$x(n) = \sum_{i=1}^{|G|} p(g_i, n) \,. \tag{4}$$

The completeness of an approximate query is the fraction of groups in the base data that are also in the sample:

$$c(n) = \frac{x(n)}{|G|}\,. \tag{5}$$

For selections, the completeness of the approximate result simply evaluates to f.

Memory Cost. The memory cost of a sample is made up of the number and the length of the tuples in the sample—as samples should be small, only required attributes are included into the sample. Hence, the absolute memory cost m_{abs} is given by $m_{abs}(n) = n \cdot l$. As for the decrease in response time, we can use the relative memory cost, which is given by:

$$m_{rel}(n) = \frac{n \cdot l}{N \cdot L}\,. \tag{6}$$

Estimation Error. Let $a_1, \ldots, a_l$ be the attributes that are aggregated in the query of interest. We now show how to compute estimation error for the AVG aggregate; the computation for the SUM aggregate is similar. For COUNT aggregates, the computation has to be adapted accordingly. We do not consider MIN or MAX as for these aggregation functions no estimation error can be computed.

Let RSD_j be the relative standard deviation of attribute a_j. Now, we can easily compute the relative standard error

$$RSE_{\hat{\mu}_j}(n) = RSD_j \sqrt{\frac{1}{n} - \frac{1}{N}} \tag{7}$$

which allows us to compare the error over multiple attributes. The overall estimation error over all the aggregation attributes is given by

$$RSE_{\hat{\mu}}(n) = \frac{1}{l} \sum_{j=1}^{l} RSE_{\hat{\mu}_j}(n)\,. \tag{8}$$

For queries with Group-By operations, the RSE is first computed for each group and then averaged over all groups. This can be done very efficiently in a single table scan by incrementally maintaining the size as well as the sum and the sum of squares of the aggregation attributes of each group [5].

Summing Up. We now put the pieces together. With the individual equations given above, the weight $w(n)$ of a sample of size n is computed by:

$$w(n) = \frac{\Delta t_{rel}(n) \cdot c(n)}{m_{rel}(n) \cdot RSE(n)}\,. \tag{9}$$

Note, this weight function is free of parameters (aside from the sample size) which was a main goal of our solution; we hold that parameters mainly confuse the user. The weight of a sample represents its effectiveness. It reflects the quality of the result and the time saving as well as the price to pay.

This weight computation, however, has one shortcoming: The maxima of different samples differ both in amplitude and position, which makes comparisons of weights impractical. As a solution, we propose the following normalization:

$$\bar{w}(n) = \frac{w\left(a \cdot n\right)}{a} \tag{10}$$

with $a = \max(w(n))$. This normalization is based on the observation that high deviations of the data—and thus, large estimation errors—result in low weights. However, in order to provide good estimates, we need large samples for data with high deviations.

Based on the proposed weight function, we next introduce our sample advisor.

2.2 Expertise-Based Sample Configuration

In an expertise-based sample configuration, our sample advisor computes for each of the query given by the expert a piece of sample advice. Such a piece of sample advice comprises the base data R, the attributes A to be included into the sample and the sample size n, thus $SA = (R, A, n)$. The first two values can easily be derived from the query; the last one is determined with the weight of a sample given above. Hence, in order to compute the sample advice for a given query q, we first set R to the base data of q and A to the set of attributes referenced by q. Then, we determine the sample size with the following two steps:

1. Scan the base data of the query once and compute for each group the relative standard deviation and the size.
2. Iterate over different sample sizes and compute the weight. Remember the sample size with the largest weight.

The effort of the first step depends on the cardinality of the base data, and thus, is fixed. The effort of the second step, however, depends on the number of regarded sample sizes. As the weight function has a single maximum it can efficiently be found by algorithms like hill climbing or binary search.

The optimal expertise-based sample configuration SC_E now consist of all the samples specified by the pieces of sample advice of the expert-given queries.

3 Extension: Workload-Based Sample Configuration

Besides relying on an expert one common approach is to utilize recorded workload information. Clearly, in such a case we cannot materialize the samples for all the queries. We thus propose the following extension of our sample advisor. Note that this extension does not result in an optimal solution but shows how the approach proposed above can easily be adapted to workload-based scenarios.

Let the workload W be a (multi-)set of queries with $W = \{q_1, \ldots, q_k\}$. Before we compute the workload-based sample configuration SC_W, we preprocess the workload by eliminating all but aggregation queries so that the workload only consists of queries relevant for approximate query processing. Further, the multiset of queries is transformed to a set by replacing the queries by (query, counter) pairs and merging duplicate queries. During this merge, predicates of the queries are ignored. [1] Hence, we get $W_{AQP} = \{(q_1, c_1), \ldots, (q_l, c_l)\}$.

With a memory constraint M, we get the following two steps:

1. Compute the optimal sample for each query of W_{AQP}. The result is a candidate set with pieces of sample advice $C = \{SA_1, \ldots, SA_k\}$.
2. Compute the optimal sample configuration SC_W of size M based on the candidate set C from the first step.

Obviously, with the first step we utilize our weight function and the computation of the expertise-based sample configuration. For the second step, we need a measure to compare different configurations. As a desirable sample configuration is characterized by a preferably low overall runtime of the workload, we use this overall runtime as our measure and try to minimize it.

As before, we assume table scans, and since we are not interested in actual response times, we simply use $r = n \cdot l$ as the response time for approximate queries and $r = N \cdot L$ for all queries with no sample in SC_W as they have to be answered with the base data. Now, the measure $\mathcal{F}$ of a sample configuration is:

$$\mathcal{F}(SC) = \sum_{q_i \in W_{AQP}} r_i \, . \tag{11}$$

The goal of the second step is to find the sample configuration that fits into M and minimizes $\mathcal{F}$. Obviously, the exact solution is an instance of the knapsack problem, which is known to be NP-hard.

Optimal Solution. To find the optimal solution (based on C), we consider all subsets C' of the candidate set C that fit into M and compute $\mathcal{F}$. The final sample configuration is the subset with the minimal measure $\mathcal{F}$.

Greedy Solution. The basic idea of the greedy approach is to successively add the most valuable candidates to the sample configuration until the memory bound is hit. We first order the candidate set C by the following score in descending order:

$$score(SA_i) = c_i \cdot \frac{N_i \cdot L_i}{n_i \cdot l_i} \, . \tag{12}$$

This score is composed of the query counter to account for the frequency of the sample usage—the more often a sample is used the more profitable gets its materialization—and the inverse of the relative memory cost. The second part of the score makes samples with smaller sampling fractions to be considered more valuable. This is motivated by the fact that those samples have low storage requirements and offer large response time benefits.

[1] We do not consider predicates as samples should be very general. Otherwise, samples would get something like materialized views.

Table 1. Sample advice candidates

Sample advice	Base data	Attributes	Sample size	Memory	Score
SA_1	R_1	$\{A_1, A_2\}$	4	8	62.5
SA_2	R_1	$\{A_3, A_4, A_5\}$	5	15	33.3
SA_3	R_1	$\{A_2, A_3, A_4\}$	7	21	23.8
SA_4	R_2	$\{A_1, A_3\}$	2	4	22.5

Next, we start with an empty sample configuration SC_W and successively add the candidates to SC_W in the given order until the next candidate does not fit into M. At this point, we allow to skip individual candidates in order to add those that still fit into M even if their score is lower.

Example 1. Consider 2 relations R_1 and R_2 with $N_1 = 100$ and $N_2 = 30$ as well as $L_1 = 5$ and $L_2 = 3$; for simplification, we say that the length of each attribute is 1 throughout the paper. With the candidates given in Table 1, let further $C = C_{ordered} = \{SA_1, SA_2, SA_3, SA_4\}$ be the (already ordered) candidate set. This table also shows the memory consumption—as defined by $n \cdot l$—and the scores of the candidates given that $c_i = 1$ for all $q_i \in W_{AQP}$ Now, let $M = 40$. We successively add the elements of $C_{ordered}$ to the initially empty SC_W. After having added SA_2, the memory consumption of SC_W is 23. Now, SA_3 does not fit into M while SA_4 does. Consequently, we skip SA_3 and add SA_4. The final sample configuration is $SC_W = \{SA_1, SA_2, SA_4\}$. The measure $\mathcal{F}$ of this sample configuration is $\mathcal{F}(SC) = 8 + 15 + 500 + 4 = 527$. □

3.1 Merging Pieces of Sample Advice

Besides the 'simple' selection of samples, we propose a second strategy that additionally considers the possibility of merging multiple pieces of sample advice. This idea is based on the following observation: In typical OLAP scenarios, many queries have the same base data—especially if the predicates are disregarded—and the referenced attributes often overlap. Hence, up to now there are samples in SC_W with the same base data and overlapping attribute sets. In order to effectively use the available memory, we consider to merge those samples. Then, the queries of both samples can be answered by the single merged sample; the redundancy in SC_W decreases.

Prerequisites for merging two pieces of sample advice SA_i and SA_j are the same base data $R_i = R_j$ as well as overlapping attributes: $A_i \cap A_j \neq \emptyset$. The merged piece of sample advice $SA_{i+j} = (R_{i+j}, A_{i+j}, n_{i+j})$ is computed by:

- $R_{i+j} = R_i = R_j$,
- $A_{i+j} = A_i \cup A_j$, and
- $n_{i+j} = \max\{n_i, n_j\}$.

When merging two pieces of sample advice, we take the maximum sample size for the following two reasons: First, decreasing n results in (considerably) higher errors (estimation error and missing tuples) and second, increasing n has less impact on $\bar{w}$ than decreasing it.

However, aside from the prerequisites, one has to verify whether or not a merge is beneficial. Clearly, a query q_i of sample advice SA_i must read $A_{i+j} \setminus A_i$ additional attributes and $n_{i+j} - n_i$ additional tuples when using the sample of SA_{i+j} instead of the sample of SA_i. Having $\mathcal{F}$ in mind, a merge is only beneficial if the overall runtime of the workload decreases, and thus, if the merge frees enough memory to add an additional sample to SC_W.

Optimal Solution. For the optimal solution, we consider all possible merges. For each considered merge, we replace the respective pieces of sample advice by the merged piece of sample advice and proceed as in the strategy without merge. Obviously, this procedure is very expensive.

Greedy Solution. With the greedy approach here, we initially proceed as in the greedy approach without merge: We order the candidate set C by the *score* value and start by adding the pieces of sample advice into an initially empty sample configuration SC_W. However, when reaching a piece of sample advice that does not fit into the memory bound M, we now try to merge individual pieces of sample advice so that the current sample advice fits into M. If there is no merge for the current sample advice, we skip this piece of sample advice and proceed with the next one until all candidates are considered or the memory bound is hit.

Greedy Merge of Sample Advice. Recall that the goal of merging sample advice is to free enough memory to add the current piece of sample advice SA_i to the sample configuration SC_W. With the greedy approach, we proceed as follows: We consider all possible merges of the sample advice currently in $SC_W \cup SA_i$. From these merges, we choose the most beneficial one, i.e., the merge that frees the most memory. In the case of equal memory consumptions, we additionally consider the overall runtime $\mathcal{F}(CS)$. If the available memory is still too low, we again look for the most beneficial merge, but this time, we replace the two pieces of sample advice chosen to merge with the merged piece of sample advice. We repeat this procedure until either enough memory is freed or no more beneficial merges are possible. In the former case, we perform the merges, in the latter case, we skip the current piece of sample advice. In the greedy merge process, the repeated procedure of finding the most beneficial merge can be done very efficiently—all we need are the sample sizes, the tuple lengths and the overlap of the merge candidates.

Example 2. Consider again the setting of Example 1. We start by adding SA_1 and SA_2 to the initially empty sample configuration SC_W. Next, SA_3 is considered. SA_3 does not fit into M and thus, we try to find a merge. Since the attribute sets of SA_1 and SA_2 are disjoint, we do not merge these pieces of sample advice. SA_3, however, meets the condition to be merged with either SA_1 or SA_2 and we determine the more beneficial merge. For both SA_{1+3} and SA_{2+3}, we have $m_{abs} = 28$ and thus, we also have to consider the overall runtime. With

SA_{1+3}, the overall runtime of q_1, q_2, and q_3 sums up to $r = 28 + 15 + 28 = 71$, while with SA_{2+3}, the overall runtime is only $r = 8 + 28 + 28 = 64$. Thus, we prefer SA_{2+3} and we have $SC_W = \{SA_1, SA_{2+3}\}$ with a memory consumption of 36. In the next step, we add SA_4 to SC_W, and our final sample configuration is $SC_W = \{SA_1, SA_{2+3}, SA_4\}$. For this sample configuration, we get $\mathcal{F}(SC) = 8 + 28 + 28 + 4 = 68$, which is significantly lower—and thus, better—than $\mathcal{F}(SC) = 527$ of the greedy approach without merging sample advice. □

4 Experiments

We ran a variety of experiments to analyze the cost model and to compare the strategies for the construction of workload-based sample configurations.[2] We compared the strategy without merging samples (*NoMerge*) with the strategy that considers the merge of samples (*Merge*). We further evaluated the effectiveness and the efficiency of the heuristic algorithms (*NoMergeGreedy* and *MergeGreedy*). We experimented with well-defined synthetic datasets in order to discover the impact of certain "data formations" on the weight function and the resulting sample configuration. Finally, we ran experiments on a large real-world dataset consisting of retail data.

Note that the considered algorithms are deterministic with respect to the resulting sample configuration. Hence, our measure $\mathcal{F}$ for the comparison of the proposed algorithms can be computed analytically.

4.1 Experimental Setup

We implemented the sample advisor on top of DB2 using Java 1.6. The experiments were conducted on an Athlon AMD XP 3000+ system running Linux with 2GB of main memory.

Cost Model. For the evaluation of our cost model, we generated a small synthetic dataset R with $N = 1,000$ tuples and $L = 10$ attributes. The specific properties of this dataset are given in Table 2. Unless stated otherwise, the parameters take the value given in the last column (Default value).

Sample Configuration. For the evaluation of the workload-based sample configurations, we used two different datasets:

- A very small synthetic dataset with $N = 100$ tuples and $L = 15$ attributes. For this dataset, we used a workload of 5 carefully chosen queries.
- A large real-world dataset of retail data. The fact table of this dataset consists of $13,223,779$ tuples with $L = 15$ attributes (5 attributes are used for grouping and 8 for aggregation). Additionally, we chose 2 of the dimension tables which also consist of a few aggregation attributes. The workload of this dataset consists of 15 typical OLAP queries.

[2] We did not evaluate the expertise-based sample configuration as its computation is trivial once the cost model is given.

Table 2. Parameters for the experiments

Parameter	**Range of values**	**Default value**
Number of groups	50 – 200	100
Skew of group sizes	0 – 1.4	0.86
Average RSD	5 – 50	15

4.2 Analysis of the Cost Model

In the first part of our experiments, we analyzed the proposed cost model. We varied several parameters of the base data and computed the weight for samples of different sizes, each with $l = 4$ attributes.

Number of Groups. In the first experiment, we varied the number of groups from 50 to 200. Figure 1a shows the impact of the number of groups on the optimal sample size: For 50 groups, we get 5 tuples as optimal sample size, for 200 groups this values increases to 38. The reason is: more groups mean smaller groups which in turn are more likely to be missing in a sample. Thus, the optimal sample size increases with increasing number of groups.

Skew of Group Sizes. Next, we varied the skew of the group sizes. We chose a Zipfian distribution with z values ranging from $z = 0$ (uniform) to $z = 1.4$ (highly skewed). Here, the value of $z = 0.86$ results in a 90-10 distribution. The result is shown in Figure 1b. As can be seen, larger skews result in larger optimal sample sizes. Again, the reason is that smaller groups are more likely to be missing: The more skewed the group sizes, the more small groups in the base data. Hence, the larger the skew, the larger the optimal sample size.

Relative Standard Deviation of Aggregation Values. Finally, we varied the relative standard deviation of the base data. Inspired by our real-world dataset, we chose values from $RSD = 5$ to $RSD = 50$. As the RSD directly influences the estimation error (see (7)), larger RSDs result in larger estimation errors and thus, in larger optimal sample sizes. This is also shown in Figure 1c.

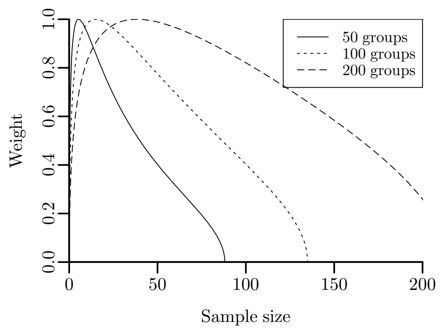
Summary. The results of the experiments show that the proposed cost model reflects the characteristics of the underlying data. Further, the advised sample sizes between about 0.5% and 5% look reasonable.

4.3 Sample Configuration

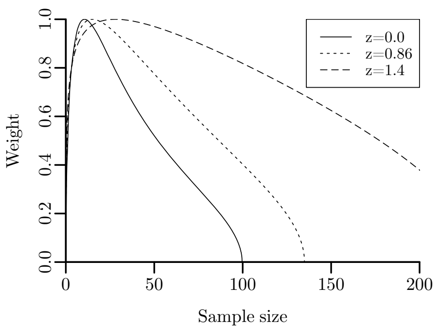
In the second part of our experiments, we compared the strategies and the algorithms for the computation of the sample configuration.

Synthetic Dataset. As stated above, we first evaluated our algorithms on a small dataset and we carefully selected 5 queries. With our cost model, we got the 5 pieces of sample advice illustrated in Figure 2 with the following scores:

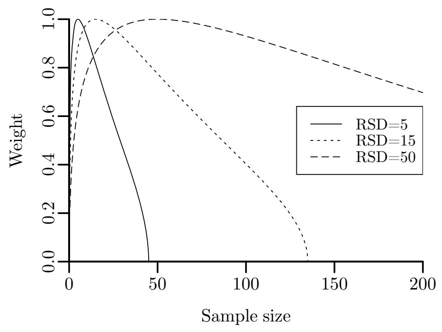
Sample advice	SA_1	SA_2	SA_3	SA_4	SA_5
Score	2035.7	2000	1900	1000	1000



(a) Number of groups



(b) Skew of group sizes



(c) Rel. standard deviation

Fig. 1. Sample weight for various settings

Next, we computed the sample configuration with all four algorithms: *NoMerge*, *NoMergeGreedy*, *Merge*, and *MergeGreedy*. We varied the memory bound from $M = 0$ to $M = 79$ attribute values (for $M = 79$, all samples fit into the memory bound) and computed $\mathcal{F}$, i.e., the runtime of the 5 queries in terms of number of read attribute values. As can be seen in Figure 3a, there are some memory bounds where the merge of samples considerably decreases $\mathcal{F}$, e.g., for $M = 53$, the merge decreases $\mathcal{F}$ from $45,620$ to $18,552$ by a factor of about 2.5. The impact of the strategy and the greedy proceeding can be seen in the close-up on $\mathcal{F}$ for $M = 20$ to $M = 45$, see Figure 3b: For $M = 25$, the sample configurations of *NoMerge* and *NoMergeGreedy* consist of only SA_2 while *Merge* and *MergeGreedy* can significantly reduce $\mathcal{F}$ by merging SA_2 and SA_3. For $M = 28$, the sample configuration of *MergeGreedy* switches from SA_{2+3} to SA_1 and thus, it even performs worse than for $M = 27$. This is a drawback of the greedy proceeding. Furthermore, for $M = 30$ *NoMerge* selects SA_2 and SA_3 and thus, results in a better sample configuration than *MergeGreedy* that still selects SA_1. These results show that for our carefully chosen queries, the effectiveness of *MergeGreedy* (temporarily) may decrease for increasing memory bounds, and that *NoMerge* may be more effective than *MergeGreedy*. All in all, *Merge* always results in the best configuration, while *NoMergeGreedy* always results in the worst.

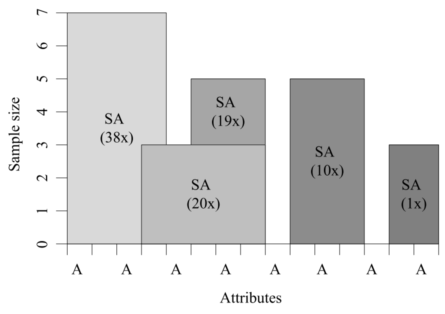
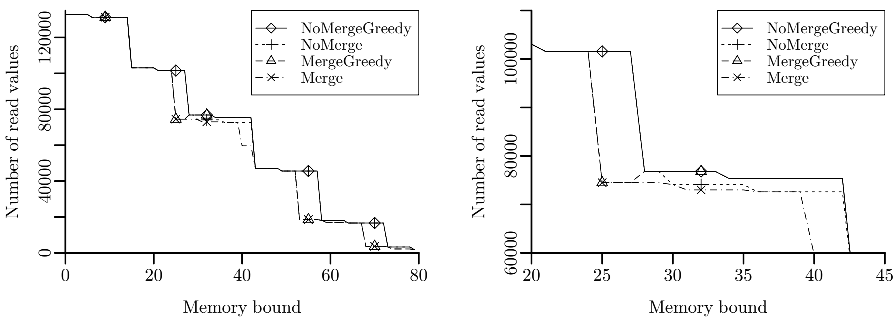


Fig. 2. Sample advice for the synthetic dataset

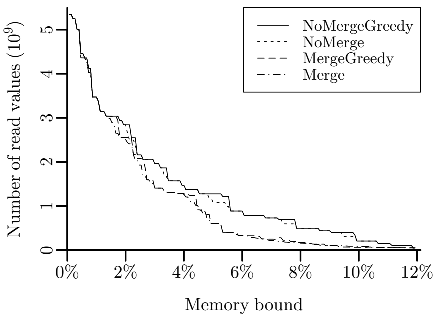


(a) Full range (b) Details

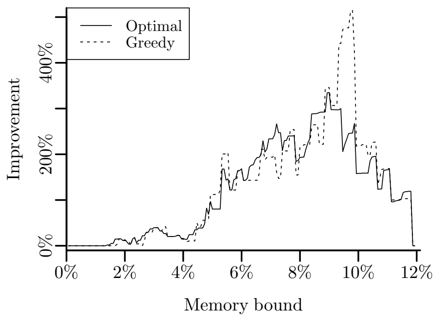
Fig. 3. Sample configurations for different memory bounds

Real-World Dataset. Our next experiments were conducted on our large real-world dataset. The effectiveness of the different algorithms is given in Figures 4a and 4b. Note that in order to make the results easier to interpret, we used relative memory bounds in the plots. Again, we computed $\mathcal{F}$ for different memory bounds, see Figure 4a. Our results show that it is beneficial to merge sample advice. Further, they clearly demonstrate the effectiveness of the greedy algorithms. To make the benefit of merging sample advice even clearer, Figure 4b depicts the improvement achieved by merging sample advice. As can be seen, the improvement quickly reaches 100%, i.e., the number of attribute values that have to be read halves due to the merges. All in all, for a broad range of memory bounds it is very beneficial to consider merges of sample advice.

In a final experiment, we compared the efficiency of our algorithms. We varied the number of candidates from $|C| = 1$ to $|C| = 15$ and computed the sample configuration with all of our algorithms. Figure 5 illustrates the time to compute the sample configurations. These values are averaged over 50 runs, and we used a relative memory bound of 6%. The plot clearly shows the benefit of the greedy



(a) Sample configurations for different memory bounds



(b) Improvement achieved by merging sample advice

Fig. 4. Real-world dataset

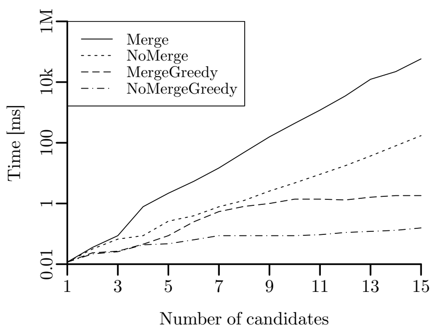


Fig. 5. Runtimes of the algorithms

proceeding: While the effort of the optimal solutions quickly gets high and exponentially increases with the number of candidates, the effort for the greedy solutions is significantly lower. For *NoMergeGreedy*, the effort is logarithmic due to the ordering of the candidates, and for *MergeGreedy*, the effort is quadratic due to the greedy merge. In an additional run, we measured the times for the greedy algorithms for $|C| = 75$ candidates. Here, the computation of the sample configuration still took less than a millisecond for *NoMergeGreedy* and about 1.3 seconds for *MergeGreedy*.

Summary. Our results on synthetic and real-world datasets show that the merge of sample advice is beneficial. It significantly reduces the runtime of the given workload. Additionally, the greedy algorithms are very efficient and effective—they often result in the same sample configurations as the exhaustive approaches while requiring only a fraction of the time to compute the configuration.

5 Related Work

In this section, we briefly review related work in the field of automatic sample selection for approximate query processing in databases which is barely studied. Some initial ideas in this field are those of [9] and [10]. The strategy of the technique shown in [9] is to select from all possible synopses those that influence the query plan or the execution costs. However, the problem of a memory bound and hence, the partition of the available space is not regarded. The solution in [10] proposes a technique for the selection of synopses as well as for the partitioning of the available memory. However, all the considerations build on spline-based synopses, so that the solutions cannot easily be translated into samples. Moreover, the focus of both solutions is the selectivity estimation where the approximation error bears another meaning as it is not directly passed to the user.

The problem is also related to the physical design problem for indexes [11] and materialized views [12]. Most of these solutions use a what-if interface [13] and ask the optimizer for the benefit. However, optimizer calls are expensive and estimated costs may be far off [14]. Moreover, the extension of the what-if interface in order to estimate both the cost and the error introduced by approximate query processing might be a complex task. The alternative solution is to define an explicit cost model as done by the approaches in [12,8]. Our weight function was inspired by that of [8] but had to be extended for both the estimation error and the incompleteness.

6 Summary

In this paper, we proposed a sample advisor for the approximate answering of analytical queries. This sample advisor is based on a novel cost model for the sample selection. We proposed a weight function that enables us to give sample advice for any individual query. Building on that, we regarded two different scenarios: (i) an expertise-based sample configuration for individually specified queries, and (ii) a workload-based sample configuration for a recorded workload regarding a given memory bound. For the computation of the workload-based sample configuration, we presented two strategies: The first strategy selects from the available sample advice those pieces of advice that minimize the runtime of the workload. The second strategy provides a more sophisticated solution by merging pieces of sample advice, which may significantly reduce the overall runtime of the given workload. For both strategies, we presented and evaluated an exact and a heuristic algorithm. Our experiments have shown that the merge of samples is almost always beneficial and provides large runtime savings for the given workload. Furthermore, our greedy algorithms significantly reduce the computation cost with only low impact on the effectiveness.

Acknowledgment. We thank Torsten Weber for his inspiring ideas.

References

1. Winter, R.: Scaling the Data Warehouse. Intelligent Enterprise (2008), http://www.intelligententerprise.com/showArticle.jhtml?articleID=211100262
2. Chaudhuri, S., Das, G., Datar, M., Narasayya, R.M.V.: Overcoming Limitations of Sampling for Aggregation Queries. In: ICDE, pp. 534–544 (2001)
3. Rösch, P., Gemulla, R., Lehner, W.: Designing Random Sample Synopses with Outliers. In: ICDE, pp. 1400–1402 (2008)
4. Acharya, S., Gibbons, P., Poosala, V.: Congressional Samples for Approximate Answering of Group-By Queries. In: SIGMOD, pp. 487–498 (2000)
5. Rösch, P., Lehner, W.: Sample Synopses for Approximate Answering of Group-By Queries. In: EDBT, pp. 403–414 (2009)
6. Acharya, S., Gibbons, P.B., Poosala, V., Ramaswamy, S.: Join Synopses for Approximate Query Answering. In: SIGMOD, pp. 275–286 (1999)
7. Gemulla, R., Rösch, P., Lehner, W.: Linked Bernoulli Synopses: Sampling Along Foreign-Keys. In: SSDBM, pp. 6–23 (2008)
8. Zilio, D.C., Zuzarte, C., Lohman, G.M., Pirahesh, H., Gryz, J., Alton, E., Liang, D., Valentin, G.: Recommending Materialized Views and Indexes with IBM DB2 Design Advisor. In: ICAC, pp. 180–188 (2004)
9. Chaudhuri, S., Narasayya, V.: Automating Statistics Management for Query Optimizers. IEEE Trans. on Knowl. and Data Eng. 13(1), 7–20 (2001)
10. König, A.C., Weikum, G.: A Framework for the Physical Design Problem for Data Synopses. In: Jensen, C.S., Jeffery, K., Pokorný, J., Šaltenis, S., Bertino, E., Böhm, K., Jarke, M. (eds.) EDBT 2002. LNCS, vol. 2287, pp. 627–645. Springer, Heidelberg (2002)
11. Chaudhuri, S., Narasayya, V.R.: An Efficient Cost-Driven Index Selection Tool for Microsoft SQL Server. In: VLDB, pp. 146–155 (1997)
12. Gupta, H., Mumick, I.S.: Selection of Views to Materialize in a Data Warehouse. IEEE Trans. on Knowl. and Data Eng. 17(1), 24–43 (2005)
13. Chaudhuri, S., Motwani, R., Narasayya, V.: Random Sampling for Histogram Construction: How much is enough? In: SIGMOD, pp. 436–447 (1998)
14. Gebaly, K.E., Aboulnaga, A.: Robustness in Automatic Physical Database Design. In: EDBT, pp. 145–156 (2008)

Estimation of the Maximum Domination Value in Multi-dimensional Data Sets

Eleftherios Tiakas, Apostolos N. Papadopoulos, and Yannis Manolopoulos

Department of Informatics, Aristotle University, 54124 Thessaloniki, Greece
{tiakas,papadopo,manolopo}@csd.auth.gr

Abstract. The last years there is an increasing interest for query processing techniques that take into consideration the dominance relationship between objects to select the most promising ones, based on user preferences. Skyline and top-k dominating queries are examples of such techniques. A skyline query computes the objects that are not dominated, whereas a top-k dominating query returns the k objects with the highest domination score. To enable query optimization, it is important to estimate the expected number of skyline objects as well as the maximum domination value of an object. In this paper, we provide an estimation for the maximum domination value for data sets with statistical independence between their attributes. We provide three different methodologies for estimating and calculating the maximum domination value, and we test their performance and accuracy. Among the proposed estimation methods, our method *Estimation with Roots* outperforms all others and returns the most accurate results.

1 Introduction

Top-k and skyline queries are two alternatives to pose preferences in query processing. In a top-k query a ranking function is required to associate a score to each object. The answer to the query is the set of k objects with the best score. A skyline query does not require a ranking function, and the result is based on preferences (minimization or maximization) posed in each attribute. The result is composed of all objects that are not dominated. For the rest of the work we deal with multidimensional points, where each dimension corresponds to an attribute. Formally, a multidimensional point $p_i = (x_{i_1}, x_{i_2}, ..., x_{i_d}) \in D$ *dominates* another point $p_j = (x_{j_1}, x_{j_2}, ..., x_{j_d}) \in D$ ($p_i \prec p_j$) when: $\forall a \in \{1, ..., d\} : x_{i_a} \leq x_{j_a} \wedge \exists b \in \{1, ..., d\} : x_{i_b} < x_{j_b}$, where d is the number of dimensions. A top-k dominating query may be seen as a combination of a top-k and a skyline query. More specifically, a top-k dominating query returns the k objects with the highest domination scores. The domination value of an object p, denoted as $dom(p)$, equals the number of objects that p dominates [12,13].

The *maximum domination value* is the number of objects dominated by the top-1 (best) object. More formally, let us assign to each item t of the data set D a score, $m(t)$, which equals the number of items that t dominates: $m(t) = |\{q \in$

	A	B	C	D	E	F	G	H	I	J
Distance to beach (m)	1000	400	600	1200	300	50	100	250	500	450
Price (euros)	60	80	90	15	65	100	50	20	40	30

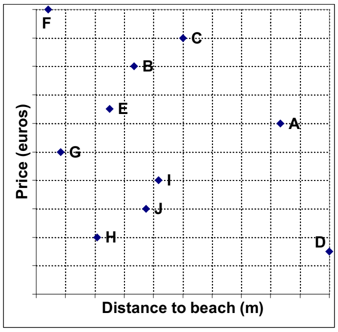


Fig. 1. The hotel data set

$D : q \prec t\}|$. Then, if p is the object with the maximum domination value we have: $p = \arg\max_t\{m(t), t \in D\}$.

An example is illustrated in Figure 1. A tourist wants to select the best hotel according to the attributes *distance to the beach* and *price per night*. The domination values of all hotels $A, B, C, D, E, F, G, H, I, J$ are 0, 1, 0, 0, 2, 0, 4, 6, 2, 3 respectively, thus the hotel with the max domination value is H. This hotel is the best possible selection, whereas the next two best choices are hotels G and J.

In this work, we focus on estimating the maximum domination value in a multi-dimensional data under the uniformity and independence assumptions. Estimating the maximum domination value contributes in: (i) optimizing top-k dominating and skyline algorithms, (ii) estimating the cost of top-k dominating and skyline queries, (iii) developing pruning strategies for these queries and algorithms. Moreover, we show that the maximum domination value is closely related to the cardinality of the skyline set.

The rest of the article is organized as follows. Section 2 briefly describes related work in the area. Section 3 studies in detail different estimation methods, whereas Section 4 contains performance evaluation results. Finally, Section 5 concludes the work.

2 Related Work

As we will show in the sequel, the maximum domination value is related to the skyline cardinality which has been studied recently. There are two different approaches for the skyline cardinality estimation problem: (i) the *parametric* methods, and (ii) the *non-parametric* methods. *Parametric* methods use only main parameters of the data set, like its cardinality N and its dimensionality d. Bentley et al. [1] established that the skyline cardinality is $O((\ln N)^{d-1})$. Buchta [3] proved another asymptotic bound of the skyline cardinality, which is:

$\Theta\left(\frac{(\ln N)^{d-1}}{(d-1)!}\right)$. Bentley et al. [1] and Godfrey [6,7], under the assumptions of attribute value independence and that all attributes in a dimension are unique and totally ordered, established that the skyline cardinality can be estimated with harmonics: $\widehat{s}_{d,N} = H_{d-1,N}$. Godfrey [6,7] established that for sufficient large N, $\widehat{s}_{d,N} = H_{d-1,N} \approx \frac{(\ln N)^{d-1}}{(d-1)!}$. Lu et al. [10] established specific parametric formulae to estimate the skyline cardinality over uniformly and arbitrary distributed data, keeping the independence assumption between dimensions.

Non-parametric methods use a sampling process in the data set to capture its characteristics and estimate the skyline cardinality. Chaudhuri et al. [4] relax the assumptions of statistical independence and attribute value uniqueness, and they use uniform random sampling in order to address correlations in the data. They assume that the skyline cardinality follows the rule: $s = A \log^B N$ for some constants A, B (which is an even more generalized formula of $\frac{(\ln N)^{d-1}}{(d-1)!}$), and using *log sampling* they calculate the A, B values. Therefore, this method can be seen as a *hybrid* method (both parametric and non-parametric). Zhang et al. [14] use a kernel-based non-parametric approach that it does not rely on any assumptions about data set properties. Using sampling over the data set they derive the appropriate kernels to efficiently estimate the skyline cardinality in any kind of data distribution.

Both directions sometimes produce significant estimation errors. Moreover, in non-parametric methods there is a tradeoff between the estimation accuracy and the sampling preprocessing cost over the data. In this paper, we focus on estimating the maximum domination value using only parametric methods. To the best of our knowledge, this is the first work studying the estimation of the maximum domination value and its relationship with the skyline cardinality.

3 Estimation Methods

In this section we present specific methods to estimate the maximum domination value of a data set. We first explain how this maximum domination value is strongly connected with the skyline cardinality of the data set. Next, we present two estimation methods inspired from [6, 7, 10], and finally we propose a novel method that is much simpler, more efficient and more accurate than its opponents.

For each presented estimation method, the main task is to produce a formula that includes only the main data set parameters, which are: the number of items of the data set (cardinality N), and the number of the existing attributes (dimensionality d). In this respect, several properties and results are derived for the maximum domination value and the item having this value. For the remaining part of this study we adopt the following assumptions:

- All attribute values in a single dimension are distinct (domain assumption).
- The dimensions are statistically independent, i.e., there are no pair-wise or group correlations nor anti-correlations (independence assumption).

Let p_i, $i \in \{1, ..., N\}$ be the N items of the data set, and (x_{i_1}, x_{i_2},..., x_{i_d}) their corresponding attributes in the d selected dimensions. Under our assumptions, no two items share a value over any dimension, thus the items can be totally ordered on any dimension. Therefore, it is not necessary to consider the actual attribute values of the items, but we can conceptually replace these values by their rank position along any dimension. Thus, let $(r_{i_1}, r_{i_2}, ..., r_{i_d})$ be the corresponding final distinct rank positions of item p_i in the selected dimensions (where $r_{i_j} \in \{1, ..., N\}$). Without loss of generality, we assume that over the attribute values in a dimension minimum is best. Then, the item with rank position 1 will have the smallest value on that dimension, whereas the item with rank position N will have the largest one.

3.1 Maximum Domination Value and Skyline Cardinality

Here we study how the maximum domination value is related to the skyline cardinality of the data set. Figure 2 reveals this relationship. Let p be the item of the data set with the maximum domination value. A first important property is that p is definitely a skyline point. This was first proved in [2] for any monotone ranking function over the data set, and also shown in [12,13] for the top-1 item in top-k dominating queries. Moreover, p dominates most of the items lying in the marked area. This area is called the *domination area* of p. No other point dominates more items than p does. Let *dom* be the exact domination value of p, which is the number of all items that lie in its domination area (i.e., the maximum domination value of the data set). On the other hand, the skyline items are the items that lie in the dotted line. Let s be the number of the skyline items (i.e., the skyline cardinality). As p does not dominate any item contained in the skyline, its domination value satisfies the relation: $dom \le N - s$. Therefore, a simple *overestimation* of the maximum domination value is $\widehat{dom} = N - s$, and can be computed when the skyline cardinality s is already known (or it has been efficiently estimated $\widehat{dom} = N - \widehat{s}$). The error rate of this estimation depends only on the items that lie neither in the skyline nor in the domination area of p, like item q for example. These items are called *outliers*. Moreover, as the data

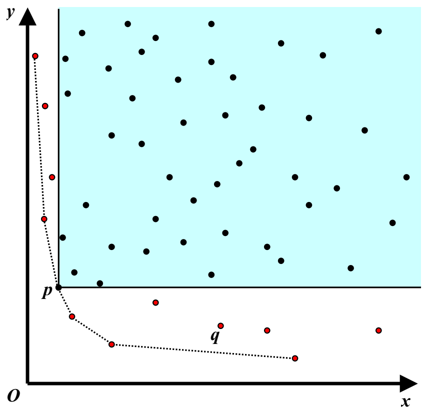


Fig. 2. Maximum domination value and skyline items

set cardinality N increases, the number of outliers becomes significantly smaller than N, and the estimation becomes more accurate. On the contrary, as the data set dimensionality d increases, the number of outliers also increases, and the estimation becomes less accurate.

3.2 Estimation with Harmonics

Here, we present an estimation approach using harmonic numbers and their properties, inspired from [6, 7]. The analysis reveals the intrinsic similarities between the maximum domination value and the skyline cardinality, and shows that $\widehat{dom} = N - \widehat{s}$. Let $dom_{d,N}$ be the random variable which measures the number of items dominated by the top-1 item (the maximum domination value). We denote as $\widehat{dom}_{d,N}$ the expected value of $dom_{d,N}$.

Theorem 1. *In any data set under the domain and independence assumptions, the expected value $\widehat{dom}_{d,N}$ satisfies the following recurrence:*

$$\widehat{dom}_{d,N} = \frac{1}{N}\widehat{dom}_{d-1,N} + \widehat{dom}_{d,N-1}$$

for $d > 1,\ N > 0$, where $\widehat{dom}_{1,N} = N - 1$ and $\widehat{dom}_{d,1} = 0$.

Proof. If $d = 1$, then we have only one dimension and the item with rank position 1 is the top-1 item that dominates all other $N - 1$ items. Thus it holds that $\widehat{dom}_{1,N} = N - 1$. If $N = 1$, then we have only one item and none item to dominate. Thus, $\widehat{dom}_{d,1} = 0$. In case that $d > 1$ and $N > 1$, there is an item with rank position 1 on dimension 1. This item has the maximum domination value as it dominates all other items on that dimension. The probability that this item will remain a top-1 item is the probability that no other item has a greater domination value in any other dimension $(2, ..., d)$, given the independence assumption. However, $\widehat{dom}_{d-1,N}$ is the maximum domination value out of these $d - 1$ dimensions. Thus, as any item has equal probability to be placed in rank position 1 on dimension 1, we have $\frac{1}{N}\widehat{dom}_{d-1,N}$ to be the probability that this item has the maximum domination value. Since, the first ranked item on dimension 1 cannot be dominated by any other item, the maximum domination value is determined by the remaining $N - 1$ items which is estimated by $\widehat{dom}_{d,N-1}$. Therefore, we have: $\widehat{dom}_{d,N} = \frac{1}{N}\widehat{dom}_{d-1,N} + \widehat{dom}_{d,N-1}$ □

The recurrence for $\widehat{dom}_{d,N}$ is strongly related to the harmonic numbers:

- The harmonic of a positive integer n is defined as: $H_n = \sum_{i=1}^{n} \frac{1}{i}$.
- The k-th order harmonic [11] of a positive integer n for integers $k > 0$ is defined as: $H_{k,n} = \sum_{i=1}^{n} \frac{H_{k-1,i}}{i}$, where $H_{0,n} = 1, \forall n > 0$ and $H_{k,0} = 0, \forall k > 0$. Note also that: $H_{1,n} = H_n, \forall n > 0$.

In order to retrieve the fundamental relation of $\widehat{dom}_{d,N}$ with the harmonic numbers, we compute $\widehat{dom}_{2,N}$ and using mathematical induction we derive the final formula. For the $\widehat{dom}_{2,N}$ value we have:

$$\widehat{dom}_{2,N} = \frac{1}{N}\widehat{dom}_{1,N} + \widehat{dom}_{2,N-1} = \frac{N-1}{N} + \widehat{dom}_{2,N-1} =$$

$$= \frac{N-1}{N} + \frac{1}{N-1}\widehat{dom}_{1,N-1} + \widehat{dom}_{2,N-2} = \frac{N-1}{N} + \frac{N-2}{N-1} + ... + \frac{1}{2} =$$

$$= 1 - \frac{1}{N} + 1 - \frac{1}{N-1} + ... + 1 - \frac{1}{2} = N - 1 - \left(\sum_{i=1}^{N}\frac{1}{i} - 1\right) = N - H_N$$

Now, let us assume that the following equation holds for a specific k, (i.e., $\widehat{dom}_{k,N} = N - H_{k-1,N}$). We will prove that the previous equation holds also for the next natural number $k+1$. We have:

$$\widehat{dom}_{k+1,N} = \frac{1}{N}\widehat{dom}_{k,N} + \widehat{dom}_{k+1,N-1} =$$

$$= \frac{1}{N}\widehat{dom}_{k,N} + \frac{1}{N-1}\widehat{dom}_{k,N-1} + \frac{1}{N-2}\widehat{dom}_{k,N-2} + ... =$$

$$= \sum_{i=1}^{N}\frac{1}{i}\widehat{dom}_{k,i} = \sum_{i=1}^{N}\frac{1}{i}\left(i - H_{k-1,i}\right) = N - \sum_{i=1}^{N}\frac{H_{k-1,i}}{i} = N - H_{k,N}$$

Therefore, for any $d > 1,\ N > 0$ it holds that:

$$\widehat{dom}_{d,N} = N - H_{d-1,N} \tag{1}$$

Equation 1 generates some important properties for the maximum domination value:

– $\widehat{dom}_{d,N}$ is strongly related to the skyline cardinality of the data set. As shown in [6,7], if $\widehat{s}_{d,N}$ is the expected value of the skyline cardinality, then it holds that: $\widehat{s}_{d,N} = H_{d-1,N}$. Therefore, we have:

$$\widehat{dom}_{d,N} = N - \widehat{s}_{d,N} \tag{2}$$

In particular, $\widehat{dom}_{d,N}$ and $\widehat{s}_{d,N}$ share the same recurrence equation of Theorem 1 but with different initial conditions.

– as proved in [11], it holds that $\lim_{d\longrightarrow\infty} H_{d,N} = N$. Therefore, we have:

$$\lim_{d\longrightarrow\infty}\widehat{dom}_{d,N} = N - \lim_{d\longrightarrow\infty}H_{d-1,N} \Leftrightarrow \lim_{d\longrightarrow\infty}\widehat{dom}_{d,N} = 0 \tag{3}$$

Equation 3 is a validation of the fact that as the dimensionality d increases, the maximum domination value (and consequently all the following domination values) decreases until reaching zero. In particular, the dimensionality d beyond which all domination values become equal to zero, is a small number. We call that dimension the *eliminating dimension*, and denote it as d_0.

In the sequel, we focus in the computation of $\widehat{dom}_{d,N}$. Since it holds that $\widehat{dom}_{d,N} = N - H_{d-1,N}$, the main task is the efficient computation of the harmonic term $H_{d-1,N}$. There are three different methods to follow for this task:

Recursive Calculation. The calculation of $H_{d-1,N}$ can be achieved by running a recursive algorithm that follows the direct definition formula:

$$H_{k,n} = \sum_{i=1}^{n} \frac{H_{k-1,i}}{i}$$

where $H_{0,n} = 1$ and $k > 0$. We can also use a look up table at run-time, however, these recurrence computations are expensive. The algorithmic time complexity is exponential: $O(N^{d-1})$. As shown later in the experimental results section, the calculation time is not acceptable even for small dimensionality values .

Bound Approximation. This method was proposed in [6, 7] and is based on asymptotic bounds of $H_{k,N}$. Bentley et al. [1] established that: $\widehat{s}_{d,N}$ is $O((\ln N)^{d-1})$. Bentley et al. [1] and Godfrey [6,7] established that: $\widehat{s}_{d,N} = H_{d-1,N}$, thus: $H_{d-1,N}$ is $O((\ln N)^{d-1})$. Buchta [3] and Godfrey [6,7] improved this asymptotic bound as follows:

$$H_{d-1,N} \approx \Theta\left(\frac{(\ln N)^{d-1}}{(d-1)!}\right)$$

Therefore, we can instantly estimate $\widehat{dom}_{d,N}$ using the following formula: (for an appropriate real number λ):

$$\widehat{dom}_{d,N} \approx N - \lambda\left(\frac{(\ln N)^{d-1}}{(d-1)!}\right) \tag{4}$$

This is not a concrete estimation and generates a significant error rate. Moreover, by varying the dimensionality range it will be shown that this estimation is not even a monotone function and changes its monotonicity after halving the eliminating dimension (i.e., for any $d > \frac{d_0}{2}$). Therefore, it provides wrong theoretical results.

Generating Functions Approximation. This method was also proposed in [6,7] and is based on Knuth's generalization via generating functions [9,8], which established that:

$$H_{k,N} = \sum_{c_1,c_2,\ldots,c_k} \prod_{i=1}^{k} \frac{\mathcal{H}_{i,N}^{c_i}}{i^{c_i} \cdot c_i!}, \qquad c_1, c_2, \ldots, c_k \geq 0 \;\wedge\; c_1+2c_2+\ldots+kc_k = k \tag{5}$$

where $\mathcal{H}_{i,N}$ is the i-th hyper-harmonic of N and is defined as: $\mathcal{H}_{i,N} = \sum_{j=1}^{N} \frac{1}{j^i}$ ($\mathcal{H}_{1,N} = H_{1,N} = H_N$).

Note that $c_1, c_2, \ldots, c_k$ are positive (or zero) integer numbers, whereas the number of terms of the sum in Equation 5 stems from all possible combinations of $c_1, c_2, ..., c_k$ that satisfy the equation $c_1 + 2c_2 + ... + kc_k = k$. This number is $\wp(k)$ and expresses the number of all possible ways to partition k as a sum of positive integers. Therefore, $H_{k,N}$ can be expressed as a polynomial of $\wp(k)$ terms which contain the first k hyper-harmonics $\mathcal{H}_{i,N}$, $(i = 1, ..., k)$. For example:

$$H_{2,N} = \frac{1}{2}\mathcal{H}_{1,N}^2 + \frac{1}{2}\mathcal{H}_{2,N} \qquad H_{3,N} = \frac{1}{6}\mathcal{H}_{1,N}^3 + \frac{1}{2}\mathcal{H}_{1,N}\mathcal{H}_{2,N} + \frac{1}{3}\mathcal{H}_{3,N}$$

This approximation of $H_{k,N}$ is remarkably accurate. In particular, with this method we reach almost exactly the theoretical values of $H_{k,N}$ when computed with the recursive approach. This will be also evaluated in the experimental results section. For any given dimension d, the time cost to compute the d required hyper-harmonics is $O(dN)$. Then, having the previous formulae, we can immediate calculate $H_{d,N}$. The only requirement is to generate the appropriate formula for the dimension d with the $\wp(d)$ terms. Godfrey [6,7] mentioned that this number of terms $(\wp(d))$ grows quickly, and, thus, it is not viable to compute the required formula this way, and suggests not using this approximation for large values of d. However, motivated by the accuracy of this approximation of $H_{k,N}$, we developed a dynamic-programming algorithm that efficiently produces these equations. Due to lack of space we do not elaborate further. Therefore, we can almost instantly estimate $\widehat{dom}_{d,N}$ with hyper-harmonics using the previous approximation formula:

$$\widehat{dom}_{d,N} \approx N - \sum_{c_1,c_2,...,c_{d-1}} \prod_{i=1}^{d-1} \frac{\mathcal{H}_{i,N}^{c_i}}{i^{c_i} \cdot c_i!} \tag{6}$$

by taking special care to all possible floating point overflow values, and by using the derived equations which recorded through the automation.

3.3 Estimation with Multiple Summations

In this section we present an estimation approach using a specific formula with multiple summations inspired from the study of [10]. For compatibility reasons we will keep all previous notations and variables.

Y. Lu et al. [10] introduced an estimation approach of the skyline cardinality that relaxes the domain assumption of our basic model. The statistical independence assumption still remains, but now the data can have duplicate attribute values. Their study is based in probabilistic methods, and it uses the value cardinality of each dimension. Their first main result is the following:

$$\widehat{s}_{d,N} = N \cdot \sum_{t_1=1}^{c_1} \sum_{t_2=1}^{c_2} ... \sum_{t_d=1}^{c_d} \left(\prod_{i=1}^{d} \frac{1}{c_i} \right) \left(1 - \prod_{j=1}^{d} \frac{t_j}{c_j} \right)^{N-1} \tag{7}$$

where $N \geq 1,\ d \geq 1$, and c_j is the value cardinality of the j-th dimension.

They also generalized this result in case of having the probability functions $f_j(x)$ of the data over each dimension, but always keeping the independence assumption:

$$\widehat{s}_{d,N} = N \cdot \sum_{t_1=1}^{c_1} \sum_{t_2=1}^{c_2} ... \sum_{t_d=1}^{c_d} f_1(t_1) f_2(t_2) ... f_d(t_d) \left(1 - \prod_{j=1}^{d} \sum_{x=1}^{t_j} f_j(x)\right)^{N-1} \tag{8}$$

However, in both cases, the computational complexity is $O(c_1 \cdot c_2 \cdot ... \cdot c_d)$, which is not acceptable even if in few dimensions the value cardinality is high (close to N). They also tried to relax this complexity cost by introducing high and low cardinality criteria, but this cost remains high, and this is why their experimental results are restricted to small dimensionality and cardinality variations ($d = 1, 2, 3$ and $N \leq 1000$). We will see in our experimental results that even if we have high cardinality in 3 dimensions and up to 1000 items the estimation time is not acceptable.

Although the method of [10] works efficiently only in small cardinalities and dimensionalities, it would be very interesting to apply this method in our model and study its accuracy. Therefore, under the domain assumption of our model, all value cardinalities c_j will be equal to N and Equation 7 gives:

$$\widehat{s}_{d,N} = N \cdot \sum_{t_1=1}^{N} \sum_{t_2=1}^{N} ... \sum_{t_d=1}^{N} \left(\frac{1}{N^d}\right) \left(1 - \frac{t_1 t_2 ... t_d}{N^d}\right)^{N-1}$$

Thus, using the property of the estimated maximum domination value of Equation 2, the final estimation formula is:

$$\widehat{dom}_{d,N} \approx N - \frac{1}{N^{d-1}} \cdot \sum_{t_1=1}^{N} \sum_{t_2=1}^{N} ... \sum_{t_d=1}^{N} \left(1 - \frac{t_1 t_2 ... t_d}{N^d}\right)^{N-1} \tag{9}$$

which has an exponential computational complexity ($O(N^d)$).

In our experimental results we will see that Equation 9 returns values remarkably close to the harmonic $H_{d-1,N}$ values. In addition, by increasing N, the returned values converge to $H_{d-1,N}$, thus it must be related somehow with the k-th order harmonics. This strong relation remains unproven. Finally, as the two methods return almost the same estimations, their accuracy is similar.

3.4 Estimation with Roots

In this section we present a novel estimation approach using a simple formula, which provides more accurate estimation results.

Let p be the item with the maximum domination value, and $(r_{p_1}, r_{p_2}, ..., r_{p_d})$ be its corresponding final rank positions in the total ordering along any dimension. Let also a be the maximum rank position of p through all dimensions (i.e., $a = \max\{r_{p_1}, r_{p_2}, ..., r_{p_d}\}$). Then, a splits the total ordering of the items in two parts as in Figure 3: (i) the (a)-area, and (ii) the ($N - a$)-area.

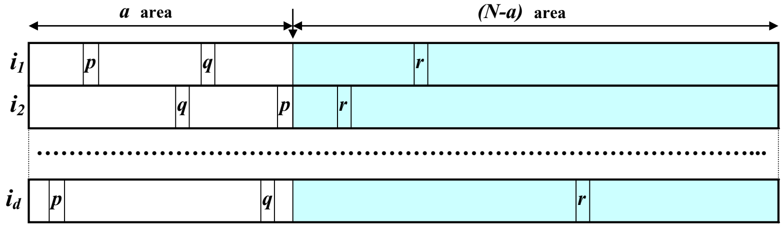


Fig. 3. Total ordering of items by rank positions

Now, any item q that all its rank positions lie in the (a)-area, will be an outlier or a skyline item. Note that the opposite does not hold, thus not any skyline or outlier item lie in the (a)-area. The probability P_a that an item p_i lies in the (a)-area is:

$$P_a = P(r_{i_1} \leq a \wedge r_{i_2} \leq a \wedge ... \wedge r_{i_d} \leq a)$$

Now, due to the independence assumption we have:

$$P_a = P(r_{i_1} \leq a) \cdot P(r_{i_2} \leq a) \cdot ... \cdot P(r_{i_m} \leq a) = \frac{a}{N} \cdot \frac{a}{N} \cdot ... \cdot \frac{a}{N} = \frac{a^d}{N^d} = \left(\frac{a}{N}\right)^d$$

Moreover, any item r that all its rank positions lie in the $(N-a)$-area, will definitely be dominated by p. Thus r lies in the domination area of p. Note that the opposite does not hold, thus not any item of the domination area of p, lies also in the $(N-a)$-area). The probability P_{N-a} that an item p_i lies in the $(N-a)$-area is:

$$P_{N-a} = P(r_{i_1} > a \wedge r_{i_2} > a \wedge ... \wedge r_{i_d} > a)$$

Due to the independence assumption we have:

$$P_{N-a} = P(r_{i_1} > a) \cdot ... \cdot P(r_{i_d} > a) = \frac{N-a}{N} \cdot ... \cdot \frac{N-a}{N} = \frac{(N-a)^d}{N^d} = \left(1 - \frac{a}{N}\right)^d$$

Therefore, the number of items lying in the (a)-area (C_a), and in the $(N-a)$-area (C_{N-a}) will be:

$$C_a = \lfloor N \cdot P_a \rfloor = \lfloor N \left(\frac{a}{N}\right)^d \rfloor \quad and \quad C_{N-a} = \lfloor N \cdot P_{N-a} \rfloor = \lfloor N \left(1 - \frac{a}{N}\right)^d \rfloor$$

respectively. However, as it holds that all items lying in the $(N-a)$-area are dominated by p, we have $dom(p) \geq C_{N-a}$, or equivalently:

$$dom(p) \geq \lfloor N \cdot P_{N-a} \rfloor \Leftrightarrow dom(p) \geq \lfloor N \left(1 - \frac{a}{N}\right)^d \rfloor \tag{10}$$

which describes a tight lower bound for the domination value of p.

Additionally, as p definitely lies in the (a)-area, at least one item must be inside that area, thus it must hold that $C_a \geq 1$, or equivalently:

$$\lfloor N \cdot P_a \rfloor \geq 1 \Leftrightarrow \lfloor N \left(\frac{a}{N}\right)^d \rfloor \geq 1 \tag{11}$$

To efficiently estimate the maximum domination value, we have to maximize the lower bound of Inequality 10 under the a variable, respecting the condition of Inequality 11 for the a variable. Therefore, let us define the function $f(a) = N\left(1 - \frac{a}{N}\right)^d$ that expresses the lower bound values, where $a \in [0, N]$. It has $f(0) = N,\ f(N) = 0$ and the following relation derives:

$$f'(a) = -d\left(1 - \frac{a}{N}\right)^{d-1}$$

We have $f'(a) < 0, \forall a \in (0, N)$, thus f is a monotone descending function in $[0,N]$, and returns values also in $[0,N]$. Moreover, the condition of Inequality 11 gives:

$$\left(\frac{a}{N}\right)^d \geq \frac{1}{N} \Leftrightarrow \frac{a}{N} \geq \sqrt[d]{\frac{1}{N}} \Leftrightarrow a \geq N\sqrt[d]{\frac{1}{N}}$$

Thus, f must be restricted in $[N\sqrt[d]{\frac{1}{N}}, N]$. Due to the descending monotonicity of f, it takes its maximum value when $a_{max} = N\sqrt[d]{\frac{1}{N}}$. Therefore, we have:

$$f(a_{max}) = N\left(1 - \frac{a_{max}}{N}\right)^d = N\left(1 - \sqrt[d]{\frac{1}{N}}\right)^d = \left(\sqrt[d]{N} - 1\right)^d$$

and the final estimation of the maximum domination value is:

$$\widehat{dom}_{d,N} \approx \left(\sqrt[d]{N} - 1\right)^d \tag{12}$$

Generalization of the Root Method. To get even more accurate estimations, we can generalize the root method by allowing the a variable to take values slightly smaller than $N\sqrt[d]{\frac{1}{N}}$. This left shift of the a variable breaks the condition of Inequality 11, but allows the possibility of taking into account items that are not lying in the $(a), (N-a)$-areas and are dominated by p increasing its domination value. We further studied this estimation improvement making exhaustive experimental tests with different a values, and we conclude that the estimation is very accurate when:

$$a_{shifted} = N\sqrt[d]{\frac{1}{N\sqrt{N}}}$$

Then f takes the value:

$$f(a_{shifted}) = N\left(1 - \frac{a_{shifted}}{N}\right)^d = N\left(1 - \sqrt[d]{\frac{1}{N\sqrt{N}}}\right)^d = \frac{1}{\sqrt{N}}\left(\sqrt[d]{N\sqrt{N}} - 1\right)^d$$

This *hidden* square root factor enhances the estimation accuracy and provides the most accurate results for the maximum domination value. Thus, the final proposed estimation formula is:

$$\widehat{dom}_{d,N} \approx \frac{1}{\sqrt{N}} \left(\sqrt[d]{N\sqrt{N}} - 1 \right)^d \tag{13}$$

4 Performance Evaluation

To test the estimation accuracy, we perform several experiments using independent data sets of N = 100, 1K, 10K, 100K, 1M items and varying the dimensionality d from 1 to values beyond the eliminating dimension d_0. We record the exact (average of 10 same type data sets) and the estimated maximum domination values. For brevity, we present only a small set of representative results, which depict the most significant aspects. All experiments have been conducted on a Pentium 4 with 3GHz Quad Core Extreme CPU, 4GB of RAM, using Windows XP. All methods have been implemented in C++. Table 1 summarizes the methods compared.

Figure 4 depicts the maximum domination value estimation results for all estimation methods, varying the cardinality and the dimensionality of the data sets. Table 2 presents the detailed estimation values of the corresponding graph for N=1K and $d = 1, ..., 10$, for further inspection. We have not recorded the values where the computational time is more than 10 minutes. Figure 5 presents

Table 1. Summary of methods evaluated

Notation	**Interpretation**
RealAvg	Real Averaged Values (No Estimation)
HarmRecc	Estimation with Harmonics (Recursive Calculation)
HarmBound	Estimation with Harmonics (Bound Approximation)
HarmGenF	Estimation with Harmonics (Generating Functions Approximation)
CombSums	Estimation with Multiple Summations
Roots	Estimation with Roots (Simple)
RootsGen	Estimation with Roots (Generalized with the square root)

Table 2. Maximum domination value estimation for $N = 1K$

d	**RealAvg**	**HarmRecc**	**HarmBound**	**HarmGenF**	**CombSums**	**Roots**	**RootsGen**
1	999.0	999.000	999.000	999.000	999.418	999.000	999.968
2	955.6	992.515	993.092	992.515	993.431	937.754	988.785
3	875.2	971.162	976.141	971.166	-	729.000	908.100
4	623.1	923.542	945.064	923.556	-	456.931	732.128
5	434.4	-	905.128	842.585	-	235.430	510.298
6	294.2	-	868.930	730.456	-	102.204	308.870
7	190.3	-	849.100	598.636	-	38.198	164.043
8	110.7	-	851.089	463.172	-	12.510	77.312
9	73.2	-	871.420	338.813	-	3.642	32.674
10	33.1	-	901.311	235.082	-	0.954	12.498

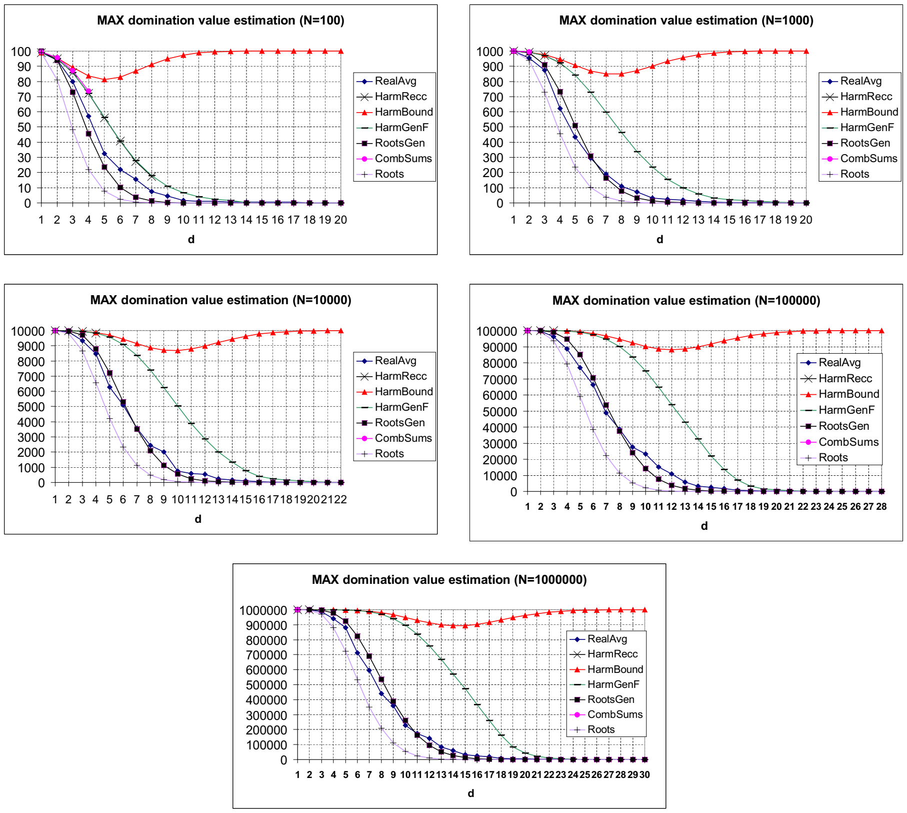


Fig. 4. Maximum domination value estimation for N=100, 1K, 10K, 100K, 1M

the estimation error of the 4 methods that return values into the full range for N=1M, for further inspection. Based on the previous results we observe the following:

- The HarmRecc method, due to the exponential computation complexity, returns values in short time only for small dimensionality and cardinality values. Moreover, after 3 dimensions the estimation error is significant.
- The HarmBound method returns estimation results instantly, but it is the most inaccurate method for estimation.
- The HarmGenF method computes its results very efficiently. It produces almost exactly the theoretical values of $H_{k,N}$ when computed recursively. Therefore, it returns the same estimation results with the HarmRecc method. However, we observe that as we increase the cardinality of the data set, the estimation error increases and becomes significant in almost all the dimensionality range.

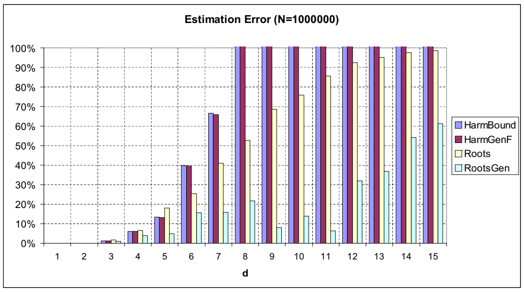


Fig. 5. Estimation error for N=1M

- The CombSums method fails to return values even in small dimensionality and cardinality selections, due to its exponential computational complexity (we can see only 4 values when N=100 and 2 values when N=1000). In addition, the returned values are remarkably close to those of HarmGenF and HarmRecc. By increasing the cardinality, the values converge to the harmonic values of the previous methods, thus it must be related somehow with the k-th order harmonics. However, again the estimation error increases and becomes significant in the whole dimensionality range.
- The Roots method returns estimation results more efficiently than all the previous methods. By increasing the cardinality, the estimation becomes more accurate, due to the fact that the number of the outliers becomes significantly smaller than N. On the contrary, as the dimensionality increases, the number of outliers increases as well, and the estimation becomes less accurate. However, when we further move into the dimensionality range and we approach the eliminating dimension, the estimation becomes again accurate.
- The RootsGen method is the most efficient way to get estimation results and outperforms all previous methods. It manages to approximate the maximum domination value with the smallest estimation error in the whole dimensionality and cardinality range.

5 Conclusions

This paper studies parametric methods for estimating the maximum domination value in multi-dimensional data sets, under the assumption of statistical independence between dimensions and the assumption that there are no duplicate attribute values in a dimension. The experimental results confirm that our proposed estimation method outperforms all other methods and achieves the highest estimation accuracy. Future work may include: (i) further study of the eliminating dimension d_0, providing an estimation formula for its calculation, (ii) the study the estimation of the skyline cardinality under the Roots method and its variants and (iii) the study of the maximum domination value estimation and the eliminating dimension in arbitrary data sets, by relaxing the assumptions of uniformity, independence and distinct values that used in this work.

References

1. Bentley, J.L., Kung, H.T., Schkolnick, M., Thompson, C.D.: On the Average Number of Maxima Set of Vectors and Applications. Journal of the ACM 25(4), 536–543 (1978)
2. Borzsonyi, S., Kossmann, D., Stocker, K.: The Skyline Operator. In: Proceedings 17th International Conference on Data Engineering (ICDE), Heidelberg, Germany, pp. 421–430 (2001)
3. Buchta, C.: On the average number of maxima in a set of vectors. Information Processing Letters 33, 63–65 (1989)
4. Chaudhuri, S., Dalvi, N., Kaushik, R.: Robust Cardinality and Cost Estimation for the Skyline Operator. In: Proceedings 22nd International Conference on Data Engineering (ICDE), Atlanta, GA (2006)
5. Huang, J.: Tuning the Cardinality of Skyline. In: Ishikawa, Y., He, J., Xu, G., Shi, Y., Huang, G., Pang, C., Zhang, Q., Wang, G. (eds.) APWeb 2008 Workshops. LNCS, vol. 4977, pp. 220–231. Springer, Heidelberg (2008)
6. Godfrey, P.: Cardinality Estimation of Skyline Queries: Harmonics in Data, Technical Report CS-2002-03, York University (2002)
7. Godfrey, P.: Skyline Cardinality for Relational Processing. In: Proceedings 3rd International Symposium of Foundations of Information and Knowledge Systems (FoIKS), pp. 78–97. Wilhelminenburg Castle, Austria (2004)
8. Graham, R.L., Knuth, D.E., Patashnik, O.: Concrete Mathematics. Addison-Wesley, Reading (1989)
9. Knuth, D.E.: Fundamental Algorithms: The Art of Computer Programming. Addison-Wesley, Reading (1973)
10. Lu, Y., Zhao, J., Chen, L., Cui, B., Yang, D.: Effective Skyline Cardinality Estimation on Data Streams. In: Bhowmick, S.S., Küng, J., Wagner, R. (eds.) DEXA 2008. LNCS, vol. 5181, pp. 241–254. Springer, Heidelberg (2008)
11. Roman, S.: The harmonic logarithms and the binomial formula. Journal of Combinatorial Theory, Series A 63, 143–163 (1993)
12. Yiu, M.L., Mamoulis, N.: Efficient Processing of Top-k Dominating Queries on Multi-Dimensional Data. In: Proceedings 33rd International Conference on Very Large Data Bases (VLDB), Vienna, Austria, pp. 483–494 (2007)
13. Yiu, M.L., Mamoulis, N.: Multi-Dimensional Top-k Dominating Queries. The VLDB Journal 18(3), 695–718 (2009)
14. Zhang, Z., Yang, Y., Cai, R., Papadias, D., Tung, A.: Kernel-Based Skyline Cardinality Estimation. In: Proceedings ACM International Conference on Management of Data (SIGMOD), pp. 509–522 (2009)

The Objects Interaction Graticule for Cardinal Direction Querying in Moving Objects Data Warehouses

Ganesh Viswanathan and Markus Schneider[⋆]

Department of Computer & Information Science & Engineering
University of Florida
Gainesville, FL 32611, USA
{gv1,mschneid}@cise.ufl.edu

Abstract. Cardinal directions have turned out to be very important qualitative spatial relations due to their numerous applications in spatial wayfinding, GIS, qualitative spatial reasoning and in domains such as cognitive sciences, AI and robotics. They are frequently used as selection criteria in spatial queries. Moving objects data warehouses can help to analyze complex multidimensional data of a spatio-temporal nature and to provide decision support. However, currently there is no available method to query for cardinal directions between spatio-temporal objects in data warehouses. In this paper, we introduce the concept of a *moving objects data warehouse* (*MODW*) for storing and querying multidimensional spatio-temporal data. Further, we also present a novel two-phase approach to model and query for cardinal directions between moving objects by using the MODW framework. First, we apply a tiling strategy that determines the zone belonging to the nine cardinal directions of each spatial object at a particular time and then intersects them. This leads to a *collection of grids* over time called the *Objects Interaction Graticule* (*OIG*). For each grid cell, the information about the spatial objects that intersect it is stored in an Objects Interaction Matrix. In the second phase, an interpretation method is applied to these matrices to determine the cardinal direction between the moving objects. These results are integrated into MDX queries using directional predicates.

1 Introduction

For more than a decade, data warehouses have been at the forefront of information technology applications as a way for organizations to effectively use information for business planning and decision making. The data warehouse contains data that gives information about a particular, decision-making subject instead of about an organization's ongoing operations (*subject-oriented*). Data is gathered into the data warehouse from a variety of sources and then merged into a coherent whole (*integrated*). All the data in a data warehouse can be identified with a particular time period (*time-variant*). Data is periodically added in a data warehouse but is hardly ever removed (*non-volatile*). This enables the manager to gain a consistent picture of the business. Thus, the *data warehouse* is a large, subject-oriented, integrated, time-variant and non-volatile collection

[⋆] This work was partially supported by the National Aeronautics and Space Administration (NASA) under the grant number NASA-AIST-08-0081.

of data in support of management's decision making process [1,2]. Online Analytical Processing (OLAP) is the technology that helps perform complex analyses over the information stored in the data warehouse. Data warehouses and OLAP enable organizations to gather overall trends and discover new avenues for growth.

With the emergence of new applications in areas such as geo-spatial, sensor, multimedia and genome research, there is an explosion of complex, spatio-temporal data that needs to be properly managed and analyzed. This data is often complex (with hierarchical, multidimensional nature) and has spatio-temporal characteristics. A good framework to store, query and mine such datasets involves *next-gen* moving objects data warehouses that bring the best tools for data management to support complex, spatio-temporal datasets. The *moving objects data warehouse* (*MODW*) can be defined as a large, subject-oriented, integrated, time-variant, non-volatile collection of analytical, *spatio-temporal* data that is used to support the strategic decision-making process for an enterprise. Moving objects data warehouses help to analyze complex multidimensional geo-spatial data exhibiting temporal variations, and provide enterprise decision support.

Qualitative relations between spatial objects include cardinal direction relations, topological relations and approximate relations. Of these cardinal directions have turned out to be very important due to their application in spatial wayfinding, qualitative spatial reasoning and in domains such as cognitive sciences, robotics, and GIS. In spatial databases and GIS they are frequently used as selection criteria in spatial queries. However, currently there is no available method to model and query for cardinal directions between moving objects (with a spatio-temporal variation).

An early approach to modeling data warehouses with support for several built-in datatypes is presented in [3]. We described a novel system to model cardinal directions between spatial regions in databases using the Object Interaction Matrix (OIM) model in [4]. This model solves the problems found in existing direction relation models like the unequal treatment of the two spatial objects as arguments of a cardinal direction relation, the use of too coarse approximations of the two spatial operand objects in terms of single representative points or MBRs, the lacking property of converseness of the cardinal directions computed, the partial restriction and limited applicability to simple spatial objects only, and the computation of incorrect results in some cases. The basis of the model was a bounded grid called the *objects interaction grid* which helps to capture the information about the spatial objects that intersect each of its cells. Then, we used a matrix to and applied an interpretation method to determine the cardinal direction between spatial objects.

In this paper, we present a novel Objects Interaction Graticule system for modeling cardinal directions between moving objects and querying for such relations. We also introduce a moving objects data warehouse framework to help achieve this task. Our method improves upon the OIM model by adding support for moving objects and provides an innovative approach to model cardinal direction relations inside data warehouses. In a first phase, we apply a multi-grid tiling strategy to determine the zone belonging to the the nine cardinal directions of each spatial object at a particular time and then intersects them. This leads to a collection of grids over time called the Objects Interaction Graticule. For each grid cell the information about the spatial objects that intersect it is stored in an Objects Interaction Matrix. In the second phase, an interpretation method is applied to these matrices to determine the cardinal direction between the moving objects. These results are integrated into MDX queries using directional predicates.

In the next section, we provide a survey of existing techniques to model cardinal directions in general, and discuss their applicability to data warehouses and for modeling direction relations between moving objects. In Section 3, we introduce our moving objects data warehouse framework and the Objects Interaction Graticule Model for modeling cardinal direction relations between moving objects. The Tiling Phase of the model (explained in Section 4) helps to generate the OIM matrix; its Interpretation is achieved in Section 5. Section 6 provides direction predicates and MDX queries [5] that illustrate cardinal direction querying using our model. Finally, Section 7 concludes the paper and provides some directions for future research.

2 Related Work

A good survey of existing approaches for modeling cardinal directions between region objects without temporal variation is provided in [4]. The models proposed to capture cardinal direction relations between simple spatial objects (like *point*, *line*, and *region* objects) as instances of *spatial data types* [6] can be classified into *tiling-based* models and *minimum bounding rectangle-based* (*MBR-based*) models, some examples of which are shown in Figure 1.

Tiling-based models use partitioning lines that subdivide the plane into tiles. They can be further classified into *projection-based* models and *cone-shaped* models, both of which assign different roles to the two spatial objects involved. The *projection-based* models define direction relations by using partitioning lines parallel to the coordinate axes. The *Direction-Relation Matrix* model [7,8] helps capture the influence of the objects' shapes as shown in Figure 1c. However, this model only applies to spatial objects with non-temporal variations. It also leads to imprecise results with intertwined objects. We introduced an improved modeling strategy for cardinal directions between region objects in [4]. The *cone-shaped* models define direction relations by using angular zones. The *MBR-based* model [9] approximates objects using minimum bounding rectangles and brings the sides of these MBRs into relation with each other using Allen's interval relations [10].

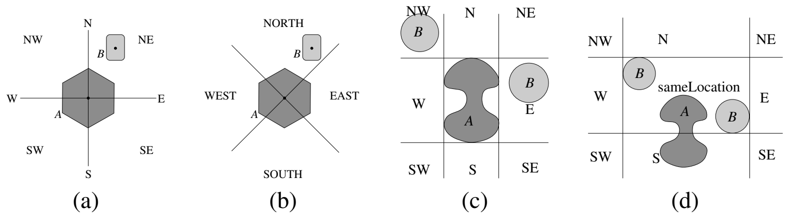


Fig. 1. Projection-based (a) and cone-shaped (b) models, and the Direction-Relation Matrix model with *A* as reference object (c) and with *B* as reference object (d)

The data warehouse helps to store and query over large, multidimensional datasets and is hence a good choice for storing and querying spatio-temporal data. We presented a conceptual, user-centric approach to data warehouse design called the *BigCube* model in [3]. Several other models have also been proposed to conceptually model a data warehouse using a cube metaphor as surveyed and extended in [11,12]. Moving objects, their formal data type characterizations and operations have been introduced in [13,14]. However there is the lack of a model for *qualitative direction relations between moving objects* in all existing works. This paper provides a clear solution to this problem by first describing the basics of the MODW framework in Section 3 and then introducing the OIG model for gathering direction relations between moving objects in Section 4.

3 Moving Objects Data Warehouses and the Objects Interaction Graticule (OIG) Approach for Modeling Cardinal Directions

The idea behind *moving objects data warehouses* (*MODW*) is to provide a system capable of representing moving entities in data warehouses and be able to ask queries about them. Moving entities could be *moving points* such as people, animals, all kinds of vehicles such as cars, trucks, air planes, ships, etc., where usually only the time-dependent position in space is relevant, not the extent. However, moving entities with an extent, e.g., hurricanes, fires, oil spills, epidemic diseases, etc., could be characterized as *moving regions*. Such entities with a continuous, spatio-temporal variation (in position, extent or shape) are called *moving objects*. With a focus on cardinal direction relations, moving regions are more interesting because of the change in the relationship between their evolving extents over time. In this paper, we focus on simple (single-component, hole-free) moving regions and provide a novel approach to gather direction relations between such objects over time, using a data warehousing framework. The *moving objects data warehouse* is defined by a conceptual cube with moving objects in the data dimensions (containing members) defining the structure of the cube, and its cells containing measure values that quantify real-world facts. The measures and members are instances of moving object data types [13]. The *BigCube* [3] is an example of a conceptual, user-centric data warehouse model that can be extended for MODW

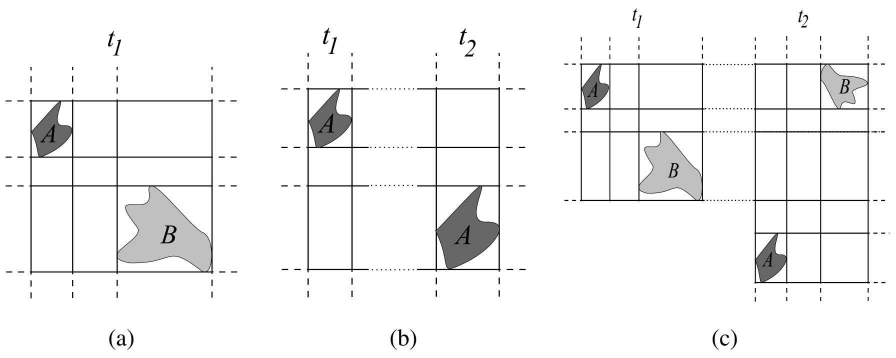


Fig. 2. Possible configurations between two objects: spatial variation *(a)*, spatio-temporal variation in one object *(b)*, spatio-temporal variation in both objects *(c)*.

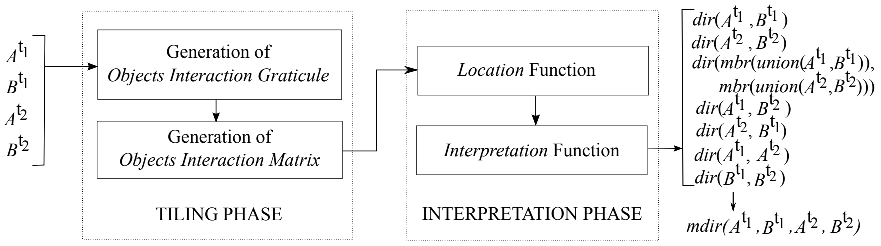


Fig. 3. Overview of the two phases of the Objects Interaction Graticule (OIG) model

design. In this paper, our OIG model lies at the conceptual level and the predicates provide the means to implement the model using any logical approach [15]. However, we shall provide MDX queries to help illustrate the versatility of the model in querying for direction relations between moving objects.

The goal of the OIG model is to enable a data warehouse user to query for cardinal directions between moving region objects. To achieve this goal, we need to take the various possible moving objects' configurations into account and model for direction relations in all of the cases to arrive at the overall direction relation. This is because the direction relation between two moving objects, between two queried time instances, can be arrived at only by considering all the direction relations between them during their *lifetimes*. The possible configurations between moving objects that we need to consider include the following. First, two objects could be at different spatial locations at the same instant of time (dual object, spatial variation) as shown in Figure 2(a). Second, an object could be at two different spatial locations at two different instances of time (single object, spatio-temporal variation) as shown in Figure 2(b). Third, two objects could be at two different spatial locations at two different time instances (dual object, spatio-temporal variation) as shown in Figure 2(c). The dotted lines between the configuration of objects across time represents the *flux* in the intersection of the coordinate systems used in the space-time continuum. We include this in our model to be able to capture the locations of objects across the time extents. However, the dashed lines indicate the part not bounded by the objects interaction graticule (OIG). The OIG is a closed, bounded region and the dotted lines do not signify any *holes* in the spatio-temporal variation of the moving objects.

Figure 3 shows the two-phase strategy of our model for calculating the cardinal direction relations between two objects A and B at time instances t_1 and t_2. We assume that A and B (in the general case) are non-empty values of the complex spatial-temporal data type *mregion* [13]. For computing the direction relation between two moving objects' snapshots, we need to consider *all possible direction relations* between the *various combination of objects* in the interacting system. First, we consider the scenario at each snapshot t_1 and t_2, and also the case when $t_1 = t_2$. For these, we default to the OIM approach for directions between objects without temporal variation and gather the direction relations between them. This is given by $dir(A^{t_1}, B^{t_1})$ and $dir(A^{t_2}, B^{t_2})$. The second case arises if $t_1 \neq t_2$. Then five more direction relations can be computed as shown in Figure 3. This includes four combinations for the two objects at t_1 and t_2, given by $dir(A^{t_1}, B^{t_2})$, $dir(A^{t_2}, B^{t_1})$, $dir(A^{t_1}, A^{t_2})$ and $dir(B^{t_1}, B^{t_2})$. Plus, we also relate the entire system (both objects) at each of the different time instances used to determine the query result. This is given by $dir(mbr(union(A^{t_1},B^{t_1})),mbr(union(A^{t_2},B^{t_2})))$. Using all these direction relations, we can now compute the moving direction relations between the two regions over time.

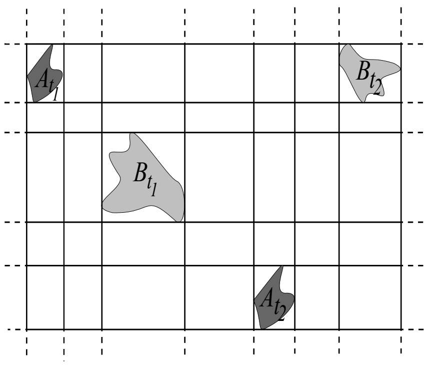
Notice that, for clarity, we have used the notation A^t instead of $A(t)$ to refer to the temporal development of the moving region A (A is actually defined by a continuous function $A : time \rightarrow region$). We will use this notation through the rest of the paper. The *tiling phase* in Section 4 details our novel tiling strategy that produces the *objects interaction graticule* and shows how they are represented by *objects interaction matrices*. The *interpretation phase* in Section 5 leverages the objects interaction matrix to derive the directional relationship between two moving region objects.

4 The Tiling Phase: Representing Interactions of Objects with the Objects Interaction Graticule and Matrix

In this section, we describe the *tiling phase* of the model. The general idea of our tiling strategy is to superimpose a graticule called *objects interaction graticule* (*OIG*) on a configuration of two moving spatial objects (regions). Such a graticule is determined by four vertical and four horizontal *partitioning lines* of *each* object at available time instances. The four vertical (four horizontal) partitioning lines of an object are given as infinite extensions of the two vertical (two horizontal) segments of the object's minimum bounding rectangle at each of the two time instances. The partitioning lines of both objects create a partition of the Euclidean plane consisting of multiple mutually exclusive, directional *tiles* or *zones*.

In the most general case, all partitioning lines are different from each other, and we obtain an overlay partition with central, bounded tiles and peripheral, unbounded tiles (indicated by the dashed segments in Figure 4 (a)). The unbounded tiles do not

$OIG(A^{t_1},B^{t_1},A^{t_2},B^{t_2}) =$



(a)

$$OIG(A^{t_1},B^{t_2}) = \begin{pmatrix} 1 & 0 & 2 \end{pmatrix}$$

$$OIM(A^{t_2},B^{t_1}) = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$OIM(mbr(union(A^{t_1},B^{t_1})), mbr(union(A^{t_2},B^{t_2}))) = \begin{pmatrix} 1 & 0 & 2 \\ 0 & 0 & 2 \end{pmatrix}$$

$$OIM(A^{t_1},A^{t_2}) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$OIM(B^{t_1},B^{t_2}) = \begin{pmatrix} 0 & 0 & 2 \\ 0 & 0 & 0 \\ 2 & 0 & 0 \end{pmatrix}$$

(b)

Fig. 4. The objects interaction graticule $OIG(A,B)$ for the two region objects A and B in Figures 1c and 1d (a) and the derived objects interaction matrices (*OIM*) for OIG components described in Definition 3

contain any objects and therefore, we exclude them and obtain a graticule space that is a bounded proper subset of $\mathbb{R}^2$, as Definition 1 states.

Definition 1. *Let $R = (A^{t_1}, B^{t_1}, A^{t_2}, B^{t_2}), R \in region$ with $A^{t_1} \neq \varnothing \wedge B^{t_1} \neq \varnothing \wedge A^{t_2} \neq \varnothing \wedge B^{t_2} \neq \varnothing$, and let $min_x^r = \min\{x \mid (x,y) \in r\}$, $max_x^r = \max\{x \mid (x,y) \in r\}$, $min_y^r = \min\{y \mid (x,y) \in r\}$, and $max_y^r = \max\{y \mid (x,y) \in r\}$ for $r \in \{A^{t_1}, B^{t_1}, A^{t_2}, B^{t_2}\}$. The* objects interaction graticule space (OIGS) *of $A^{t_1}, B^{t_1}, A^{t_2}$ and B^{t_2} is given as:*

$$\begin{aligned} OIGS(R) = \{(x,y) \in \mathbb{R}^2 \mid & \min(min_x^{A^{t_1}}, min_x^{B^{t_1}}, min_x^{A^{t_2}}, min_x^{B^{t_2}}) \leq x \leq \\ & \max(max_x^{A^{t_1}}, max_x^{B^{t_1}}, max_x^{A^{t_2}}, max_x^{B^{t_2}}) \wedge \min(min_y^{A^{t_1}}, min_y^{B^{t_1}}, \\ & min_y^{A^{t_2}}, min_y^{B^{t_2}}) \leq y \leq \max(max_y^{A^{t_1}}, max_y^{B^{t_1}}, max_y^{A^{t_2}}, max_y^{B^{t_2}})\} \end{aligned}$$

Definition 2 determines the bounded graticule formed as a part of the partitioning lines and superimposed on $OIGS(A^{t_1}, B^{t_1}, A^{t_2}, B^{t_2})$.

Definition 2. *Let* seg *be a function that constructs a segment between any two given points $p, q \in \mathbb{R}^2$, i.e., $seg(p,q) = \{t \mid t = (1-\lambda)p + \lambda q, 0 \leq \lambda \leq 1\}$. Let $H_r = \{seg((min_x^r, min_y^r), (max_x^r, min_y^r)), seg((min_x^r, max_y^r), (max_x^r, max_y^r))\}$ and $V_r = \{seg((min_x^r, min_y^r), (min_x^r, max_y^r)), seg((max_x^r, min_y^r), (max_x^r, max_y^r))\}$ for $r \in \{A^{t_1}, B^{t_1}, A^{t_2}, B^{t_2}\}$. We call the elements of $H_{A^{t_1}}$, $H_{B^{t_1}}$, $H_{A^{t_2}}$, $H_{B^{t_2}}$, $V_{A^{t_1}}$, $V_{B^{t_1}}$, $V_{A^{t_2}}$ and $V_{B^{t_2}}$* objects interaction graticule segments. *Then, the* objects interaction graticule (OIG) *for A and B is given as:*

$$OIG(A^{t_1}, B^{t_1}, A^{t_2}, B^{t_2}) = H_{A^{t_1}} \cup V_{A^{t_1}} \cup H_{B^{t_1}} \cup V_{B^{t_1}} \cup H_{A^{t_2}} \cup V_{A^{t_2}} \cup H_{B^{t_2}} \cup V_{B^{t_2}}.$$

In the OIG of an object, there are two constituent *object interaction coordinate systems* (*OICS*) for each temporal state of the moving objects. These are defined as follows:

$$OICoordS(A^{t_1}, B^{t_1}) = H_{A^{t_1}} \cup V_{A^{t_1}} \cup H_{B^{t_1}} \cup V_{B^{t_1}}, \text{ and}$$
$$OICoordS(A^{t_2}, B^{t_2}) = H_{A^{t_2}} \cup V_{A^{t_2}} \cup H_{B^{t_2}} \cup V_{B^{t_2}}.$$

The definition of OIG comprises the description of all graticules that can arise. In the most general case, if $t_1 = t_2$ and $H_{A^{t_1}} \cap H_{B^{t_1}} = \varnothing$ and $V_{A^{t_1}} \cap V_{B^{t_1}} = \varnothing$, we obtain a bounded 3×3-graticule similar to that for a non-temporal variation in the objects configurations. Special cases arise if $H_{A^{t_1}} \cap H_{B^{t_1}} \neq \varnothing$ and/or $V_{A^{t_1}} \cap V_{B^{t_1}} \neq \varnothing$. Then equal graticule segments coincide in the union of all graticule segments. As a result, depending on the relative position of two objects to each other, the objects interaction graticule can be of different sizes. However, due to the non-empty property of a region object, not all graticule segments can coincide. This means that at least two horizontal graticule segments and at least two vertical graticule segments must be maintained. Definition 3 gives a formal characterization for the OIG.

Definition 3. *An objects interaction graticule* OIG$(A^{t_1}, B^{t_1}, A^{t_2}, B^{t_2})$ *consists of two objects interaction coordinate systems, at t_1 and t_2, each of size $m \times n$, with $m, n \in \{1,2,3\}$, if $|H_A \cap H_B| = 3 - m$ and $|V_A \cap V_B| = 3 - n$. Further, it also consists of four Objects Interaction Grids for each of the spatio-temporal combinations of the two moving objects and a fifth for the overall system. Together, these are called the* objects interaction graticule components.

The objects interaction graticule partitions the objects interaction graticule space into *objects interaction graticule tiles* (*zones*, *cells*). Definition 4 provides their definition for each of the time instances uniquely, using the objects interaction coordinate systems.

Definition 4. *Given* $A^{t_1}, B^{t_1}, A^{t_2}, B^{t_2} \in$ *region with* $A^{t_1} \neq \varnothing \wedge B^{t_1} \neq \varnothing \wedge A^{t_2} \neq \varnothing \wedge B^{t_2} \neq \varnothing$, $\mathrm{OIGS}(A^{t_1}, B^{t_1}, A^{t_2}, B^{t_2})$*, and* $\mathrm{OIG}(A^{t_1}, B^{t_1}, A^{t_2}, B^{t_2})$*, we define* $c_H = |H_A \cup H_B| = |H_A| + |H_B| - |H_A \cap H_B|$ *and* c_V *correspondingly at a time instant. Let* $H_{AB} = H_A \cup H_B = \{h_1, \ldots, h_{c_H}\}$ *such that (i)* $\forall 1 \leq i \leq c_H : h_i = seg((x_i^1, y_i), (x_i^2, y_i))$ *with* $x_i^1 < x_i^2$*, and (ii)* $\forall 1 \leq i < j \leq c_H : h_i < h_j$ *(we say that* $h_i < h_j :\Leftrightarrow y_j < y_i$*). Further, let* $V_{AB} = V_A \cup V_B = \{v_1, \ldots, v_{c_V}\}$ *such that (i)* $\forall 1 \leq i \leq c_V : v_i = seg((x_i, y_i^1), (x_i, y_i^2))$ *with* $y_i^1 < y_i^2$*, and (ii)* $\forall 1 \leq i < j \leq c_V : v_i < v_j$ *(we say that* $v_i < v_j :\Leftrightarrow x_i < x_j$*).*

Next, we define four auxiliary predicates that check the position of a point (x, y) *with respect to a graticule segment:*

$$\begin{aligned}
below((x,y),h_i) &\Leftrightarrow x_i^1 \leq x \leq x_i^2 \ \wedge\ y \leq y_i \\
above((x,y),h_i) &\Leftrightarrow x_i^1 \leq x \leq x_i^2 \ \wedge\ y \geq y_i \\
right\_of((x,y),v_i) &\Leftrightarrow y_i^1 \leq y \leq y_i^2 \ \wedge\ x \geq x_i \\
left\_of((x,y),v_i) &\Leftrightarrow y_i^1 \leq y \leq y_i^2 \ \wedge\ x \leq x_i
\end{aligned}$$

An objects interaction graticule tile $t_{i,j}$ *with* $1 \leq i < c_H$ *and* $1 \leq j < c_V$ *is then defined for a particular time instant as*

$$\begin{aligned}
t_{i,j} = \{(x,y) \in \mathrm{OIGS}(A,B) \mid\ & below((x,y),h_i) \ \wedge\ above((x,y),h_{i+1}) \ \wedge \\
& right\_of((x,y),v_j) \ \wedge\ left\_of((x,y),v_{j+1})\}
\end{aligned}$$

The definition indicates that all tiles are bounded and that two adjacent tiles share their common boundary. Let $OIGT(A,B)$ be the set of all tiles $t_{i,j}$ imposed by $OIG(A,B)$ on $OIGS(A,B)$. An $m \times n$-graticule contains $m \cdot n$ bounded tiles.

By applying our tiling strategy, an objects interaction graticule can be generated for any two region objects A and B. It provides us with the valuable information which region object intersects which tile across the temporal variations. With each time event t_1 and t_2, Definition 5 provides us with a definition of the *interaction* of A and B with a tile.

Definition 5. *Given* $A, B \in$ *region with* $A \neq \varnothing$ *and* $B \neq \varnothing$ *and* $\mathrm{OIGT}(A,B)$*, let* ι *be a function that encodes the* interaction *of* A *and* B *with a tile* $t_{i,j}$*, and checks whether no region, A only, B only, or both regions intersect a tile. We define this function as*

$$\iota(A,B,t_{i,j}) = \begin{cases} 0 \text{ if} & A^\circ \cap t_{i,j}^\circ = \varnothing \ \wedge\ B^\circ \cap t_{i,j}^\circ = \varnothing \\ 1 \text{ if} & A^\circ \cap t_{i,j}^\circ \neq \varnothing \ \wedge\ B^\circ \cap t_{i,j}^\circ = \varnothing \\ 2 \text{ if} & A^\circ \cap t_{i,j}^\circ = \varnothing \ \wedge\ B^\circ \cap t_{i,j}^\circ \neq \varnothing \\ 3 \text{ if} & A^\circ \cap t_{i,j}^\circ \neq \varnothing \ \wedge\ B^\circ \cap t_{i,j}^\circ \neq \varnothing \end{cases}$$

We use the mbr and union functions for computing the minimum bounding rectangle and the spatial union of two objects, respectively. To support both objects interaction coordinate systems we extend ι to accept $mbr(union(A^{t_1}, B^{t_1}))$ and $mbr(union(A^{t_2}, B^{t_2}))$ as operands. The operator $^\circ$ denotes the point-set topological *interior* operator and yields a region without its boundary. For each graticule cell $t_{i,j}$ in the ith row and jth column of an $m \times n$-graticule with $1 \leq i \leq m$ and $1 \leq j \leq n$, we store the coded information in an *objects interaction matrix* (*OIM*) in cell $OIM(A,B)_{i,j}$.

$$OIM(A,B) = \begin{pmatrix} \iota(A,B,t_{1,1}) & \iota(A,B,t_{1,2}) & \iota(A,B,t_{1,3}) \\ \iota(A,B,t_{2,1}) & \iota(A,B,t_{2,2}) & \iota(A,B,t_{2,3}) \\ \iota(A,B,t_{3,1}) & \iota(A,B,t_{3,2}) & \iota(A,B,t_{3,3}) \end{pmatrix}$$

5 The Interpretation Phase: Assigning Semantics to the Objects Interaction Matrix

The second phase of the OIG model is the *interpretation phase*. This phase takes an objects interaction matrix (OIM)obtained as the result of the tiling phase as input and uses it to generate a set of cardinal directions as output. This is achieved by separately identifying the locations of both objects in the objects interaction matrix and by pairwise interpreting these locations in terms of cardinal directions. The union of all these cardinal directions is the result. This phase is similar to the Interpretation Phase of the OIM model [4].

We use an interpretation function to determine the basic cardinal direction between any two object components on the basis of their (i, j)-locations in the objects interaction matrix. The composite cardinal relation between A and B is then the union of all determined relations.

In a first step, we define a function *loc* (see Definition 6) that acts on one of the region objects A or B and their OIM and determines all locations of components of each object in the matrix for both temporal extents individually. Let $I_{m,n} = \{(i, j) \mid 1 \leq i \leq m, 1 \leq j \leq n\}$. We use an index pair $(i, j) \in I_{m,n}$ to represent the location of the element $M_{i,j} \in \{0,1,2,3\}$ and thus the location of an object component from A or B in an $m \times n$ objects interaction matrix.

Definition 6. *Let M be the $m \times n$-objects interaction matrix of two region objects A and B. Then the function loc is defined as:*

$$loc(A,M) = \{(i, j) \mid 1 \leq i \leq m, 1 \leq j \leq n, M_{i,j} = 1 \ \vee \ M_{i,j} = 3\}$$
$$loc(B,M) = \{(i, j) \mid 1 \leq i \leq m, 1 \leq j \leq n, M_{i,j} = 2 \ \vee \ M_{i,j} = 3\}$$

In a second step, we define an *interpretation function* ψ to determine the cardinal direction between any two object components of A and B on the basis of their locations in the objects interaction matrix. We use a popular model with the nine *basic cardinal directions*: *north* (*N*), *northwest* (*NW*), *west* (*W*), *southwest* (*SW*), *south* (*S*), *southeast* (*SE*), *east* (*E*), *northeast* (*NE*), and *origin* (*O*) to symbolize the possible cardinal directions between *object components*. A different set of basic cardinal directions would lead to a different interpretation function and hence to a different interpretation of index pairs. Definition 7 provides the interpretation function ψ with the signature $\psi : I_{m,n} \times I_{m,n} \rightarrow CD$.

Definition 7. *Given $(i, j), (i', j') \in I_{m,n}$, the* interpretation function ψ *on the basis of the set $CD = \{\mathrm{N, NW, W, SW, S, SE, E, NE, O}\}$ of* basic cardinal directions *is defined as*

$$\psi((i,j),(i',j')) = \begin{cases} \mathrm{N} & \mathit{if}\ i < i' \wedge j = j' \\ \mathrm{NW} & \mathit{if}\ i < i' \wedge j < j' \\ \mathrm{W} & \mathit{if}\ i = i' \wedge j < j' \\ \mathrm{SW} & \mathit{if}\ i > i' \wedge j < j' \\ \mathrm{S} & \mathit{if}\ i > i' \wedge j = j' \\ \mathrm{SE} & \mathit{if}\ i > i' \wedge j > j' \\ \mathrm{E} & \mathit{if}\ i = i' \wedge j > j' \\ \mathrm{NE} & \mathit{if}\ i < i' \wedge j > j' \\ \mathrm{O} & \mathit{if}\ i = i' \wedge j = j' \end{cases}$$

The main difference compared to the OIM approach however is in the following third and final step. We temporally *lift* the *dir* cardinal direction relation function to include objects over their temporal extents. Here, we specify the *cardinal direction function* named *mdir* (moving-direction) which determines the *composite moving cardinal direction* for two moving region objects A and B. This function has the signature $mdir : region_{t_1} \times region_{t_2} \rightarrow 2^{CD}$ and yields a set of basic cardinal directions as its result. In order to compute the function *dir*, we first generalize the signature of our interpretation function ψ to $\psi : 2^{I_{m,n}} \times 2^{I_{m,n}} \rightarrow 2^{CD}$ such that for any two sets $X, Y \subseteq I_{m,n}$ holds: $\psi(X,Y) = \{\psi((i,j),(i',j')) \mid (i,j) \in X, (i',j') \in Y\}$. We are now able to specify the cardinal direction function *mdir* in Definition 8.

Definition 8. *Let* $A, B \in region$ *and* $dir(A,B) = \psi(loc(A, OIM(A,B)), loc(B, OIM(A,B)))$. *Then the* cardinal direction function mdir *is defined as*

$$\begin{aligned} mdir(A^{t_1}, B^{t_1}, A^{t_2}, B^{t_2}) = {} & dir(A^{t_1}, B^{t_1}) \cup dir(A^{t_2}, B^{t_2}) \cup dir((A^{t_1}, B^{t_1}), (A^{t_2}, B^{t_2})) \cup \\ & dir(A^{t_1}, B^{t_2}) \cup dir(A^{t_2}, B^{t_1}) \cup dir(A^{t_1}, A^{t_2}) \cup dir(B^{t_1}, B^{t_2}) \end{aligned}$$

We apply this definition to our example in Figure 4. With $loc(A^{t_1}, OIM(At_1, Bt_1)) = \{(1,1)\}$ and $loc(B^{t_1}, OIM(A^{t_1}, B^{t_1})) = \{(3,3)\}$, and so on, we obtain

$$\begin{aligned} mdir(A^{t_1}, B^{t_1}, A^{t_2}, B^{t_2}) &= \{\psi((1,1),(3,3)), \psi((3,1),(1,3)), \psi(\{(1,1)\}, \\ &\quad \{(1,3),(2,3)\}), \psi((1,1),(1,3)), \psi((3,1),(1,2)), \\ &\quad \psi((1,1),(3,1)), \psi((3,1),(1,3))\} \\ &= \{NW, SW, W, SE\} \end{aligned}$$

Finally we can say regarding Figure 4 that "Object A is *partly northwest*, *partly southwest*, *partly west*, and *partly southeast* of object B over the period from time t_1 to t_2". Each of the individual directions between the moving objects for the three possible configurations described in Section 3 can also be provided by using the results from each application of *dir*, that is used to finally arrive at the moving direction relations (given by *mdir*).

6 Directional Predicates for OLAP Querying in Moving Object Data Warehouses

Based on the OIG model and the interpretation mechanism described in the previous sections, we can identify the cardinal directions between any given two moving region objects. To integrate the cardinal directions into moving object data warehouses as selection and join conditions in spatial queries, binary *directional predicates* need to be formally defined. For example, a query like "Find all hurricanes that affect states which lie strictly to the north of Florida" requires a directional predicate like *strict_north* as a selection condition of a spatial join.

The *mdir* function, which produces the final moving cardinal directions between two complex region objects A and B across temporal variation, yields a subset of the set $CD = \{N, NW, W, SW, S, SE, E, NE, O\}$ of *basic cardinal directions*. As a result, a total number of $2^9 = 512$ cardinal directions can be identified. Therefore, at most 512 directional predicates can be defined to provide an *exclusive* and *complete* coverage of all possible directional relationships. We can assume that users will not be interested in such a large, overwhelming collection of detailed predicates since they will find it difficult to distinguish, remember and handle them. Instead we provide a mechanism for the user to define and maintain several levels of predicates for querying. As a first step, in Definition 9, we propose nine *existential directional predicates* that ensure the existence of a particular basic cardinal direction between parts of two region objects A and B.

Definition 9. *Let* $R = (A^{t_1}, B^{t_1}, A^{t_2}, B^{t_2}), R \in region$. *Then the existential directional predicate for north is defined as:*

$$exists\_north(R) \;\equiv\; (N \in mdir(R))$$

Eight further existential direction predicates for S, E, W, O, NE, SE, NW, and SW are also defined correspondingly. Later, by using $\neg$, $\vee$ and $\wedge$ operators, the user will be able to define any set of composite *derived directional predicates* from this set for their own applications.

We shall provide two examples for these. The first set of predicates is designed to handle *similarly oriented directional predicates* between two regions. *Similarly oriented* means that several cardinal directions facing the same general orientation belong to the same group. Definition 10 shows an example of *northern* by using the existential predicates.

Definition 10. *Let* $R = (A^{t_1}, B^{t_1}, A^{t_2}, B^{t_2}), R \in region$. *Then* northern *is defined as:*

$$northern(R) = exists\_north(R) \vee exists\_northwest(R) \vee exists\_northeast(R)$$

The other similarly oriented directional predicates *southern*, *eastern*, and *western* are defined in a similar way.

The second set of predicates is designed to handle *strict directional predicates* between two region objects. *Strict* means that two region objects are in exactly one basic cardinal direction to each other. Definition 11 shows an example of *strict_north* by using the existential predicates.

Definition 11. *Let* $R = (A^{t_1}, B^{t_1}, A^{t_2}, B^{t_2}), R \in region$. *Then* strict_north *is defined as:*

$$\begin{aligned} strict\_north(R) = {} & exists\_north(R) \wedge \neg exists\_south(R) \wedge \neg exists\_west(R) \wedge \\ & \neg exists\_east(R) \wedge \neg exists\_northwest(R) \wedge \quad \neg exists\_northeast(R) \wedge \\ & \neg exists\_southwest(R) \wedge \neg exists\_southeast(R) \wedge \neg exists\_origin(R) \end{aligned}$$

The other strict directional predicates *strict_south*, *strict_east*, *strict_west*, *strict_origin*, *strict_northeast*, *strict_northwest*, *strict_southeast*, *strict_southwest*, *strict_northern*, *strict_southern*, *strict_eastern*, and *strict_western* are defined in a similar way.

We can now employ these predicates in MDX queries in the moving objects data warehouse. For example, assuming we are given a sample *WeatherEvents* cube (analogous to a spreadsheet table) with hurricane names (ordered in categories according to their intensity) from several years and containing geographic information, we can pose the following query:

Determine the names of hurricanes, ordered in categories according to their intensity, which had a path moving towards the east from their point of origin, and affected states strictly in the northern part of Florida, during the period from 2005 to 2009.

The corresponding MDX query is as follows:

```
SELECT { [Date].[2005] : [Date].[2009] } ON ROWS,
{ NON EMPTY Filter( {[Measures].[Hurricanes].[Category].MEMBERS},
   exists_east( [Measures].[Hurricanes].CurrentMember,
  [Measures].[Hurricanes])) } ON COLUMNS,
{ [Geography].[Country].[State]} ON PAGES,
FROM Cube_WeatherEvents
WHERE ( strict_northern( [Geography].[Country].[State].MEMBERS,
        [Geography].[Country].[USA].[FL] ))
```

A sample result of this query is shown below.

		2005	2006	2007	2008	2009
Georgia	Cat-2	Kevin	Bronco		Tracy	Nobel
	Cat-3	Cindy	Alberto	Barry	Fay	Ida
	Cat-4	Katrina	Alberto			Ida
North Carolina	Cat-2	Cindy	Alberto		Hought	Jives
	Cat-3	Katrina	Ernesto	Sabley	Hanna	Vorice

7 Conclusion and Future Work

In this paper, we introduce the concept of a *moving objects data warehouse* (MODW) for storing and querying multidimensional, spatial-temporal data. We also present a novel approach called the *Objects Interaction Graticule* (*OIG*) model to determine the cardinal directions between moving, simple (single-component, hole-free) region objects. We also show how directional predicates can be derived from the cardinal directions and use them in MDX queries.

In the future, we plan to extend our approach to include complex moving points, lines and other mixed combinations of moving object data types. Further work includes an efficient implementation of the moving objects data warehouse, and the design of spatial reasoning techniques for direction relations using the objects interaction graticule model.

References

1. Inmon, W.: Building the Data Warehouse. John Wiley & Sons, New York (2005)
2. Kimball, R., Ross, M.: The Data Warehousing Toolkit. John Wiley& Sons, New York (1996)
3. Viswanathan, G., Schneider, M.: BigCube: A Metamodel for Managing Multidimensional Data. In: Proceedings of the 19th International Conference on Software Engineering and Data Engineering (SEDE), pp. 237–242 (2010)
4. Chen, T., Schneider, M., Viswanathan, G., Yuan, W.: The Objects Interaction Matrix for Modeling Cardinal Directions in Spatial Databases. In: Proceedings of the 15th International Conference on Database Systems for Advanced Applications (DASFAA), pp. 218–232 (2010)
5. Microsoft Corporation: Multidimensional Expressions (MDX) Reference, http://msdn.microsoft.com/en-us/library/ms145506.aspx (ccessed: June 6, 2010)
6. Schneider, M.: Spatial Data Types for Database Systems - Finite Resolution Geometry for Geographic Information Systems. In: Schneider, M. (ed.) Spatial Data Types for Database Systems. LNCS, vol. 1288. Springer, Heidelberg (1997)
7. Goyal, R., Egenhofer, M.: Cardinal Directions between Extended Spatial Objects (2000) (unpublished manuscript)
8. Skiadopoulos, S., Koubarakis, M.: Composing Cardinal Direction Relations. Artificial Intelligence 152(2), 143–171 (2004)
9. Papadias, D., Egenhofer, M.: Algorithms for Hierarchical Spatial Reasoning. GeoInformatica 1(3), 251–273 (1997)
10. Allen, J.F.: Maintaining Knowledge about Temporal Intervals. Journal of the Association for Computing Machinery 26(11), 832–843 (1983)
11. Pedersen, T., Jensen, C., Dyreson, C.: A Foundation for Capturing and Querying Complex Multidimensional Data. Information Systems 26(5), 383–423 (2001)
12. Malinowski, E., Zimanyi, E.: Spatial Hierarchies and Topological Relationships in the Spatial MultiDimER Model. In: Jackson, M., Nelson, D., Stirk, S. (eds.) BNCOD 2005. LNCS, vol. 3567, pp. 17–28. Springer, Heidelberg (2005)
13. Guting, R., Bohlen, M., Erwig, M., Jensen, C., Lorentzos, N., Schneider, M., Vazirgiannis, M.: A foundation for representing and querying moving objects. ACM Transactions on Database Systems (TODS) 25(1), 42 (2000)
14. Lema, C., Antonio, J., Forlizzi, L., Guting, R., Nardelli, E., Schneider, M.: Algorithms for Moving Objects Databases. The Computer Journal 46(6), 680 (2003)
15. Vassiliadis, P., Sellis, T.: A Survey of Logical Models for OLAP Databases. SIGMOD Record 28(4), 64–69 (1999)

Opening the Knowledge Tombs - Web Based Text Mining as Approach for Re-evaluation of Machine Learning Rules

Milan Zorman[1,2], Sandi Pohorec[1], and Boštjan Brumen[1]

[1] University of Maribor, Faculty of Electrical Engineering and Computer Science, Smetanova ulica 17, 2000 Maribor, Slovenia
[2] Centre for Interdisciplinary and Multidisciplinary Research and Studies of the University of Maribor, Krekova ulica 2, 2000 Maribor, Slovenia
{milan.zorman,sandi.pohorec,brumen}@uni-mb.si

Abstract. Growth of internet usage and content provides us with large amounts of free text information, which could be used to extend our data mining capabilities and to collect specialist knowledge from different reliable sources.

In this paper we explore the possibility for a reuse of 'old' data mining results, which seemed to be well exploited at the time of their formation, but are now laying stored in so called knowledge tombs. By using the web based text mined knowledge we are going to verify knowledge, gathered in the knowledge tombs.

We focused on re-evaluation of rules, coming from symbolic machine learning (ML) approaches, like decision trees, rough sets, association rules and ensemble approaches.

The knowledge source for ML rule evaluation is the web based text mined knowledge, aimed to complement and sometimes replace the domain expert in the early stages.

Keywords: knowledge tombs, data mining, re-evaluation, rules, supervised machine learning.

1 Introduction

In this paper we explore the possibility to begin a reuse of 'old' data mining results, which seemed to be well exploited at the time of their formation, but are now nothing else than knowledge tombs.

Although the data mining community collectively uses and attacks the data tombs created by different institutions and individuals, it also creates vast amounts of potential knowledge, which, considering the available capabilities, may seemed to be well used and exploited at the time of the research.

But is that still true after 5, 10, 15 or 20 years? What seemed to be well exploited basis, may now represent a valuable ore for further data and knowledge mining.

Just ask yourself or the first data mining researcher you run into, how many chunks of information/rules/decision trees/... you/he produced in the last decade. The answers will easily reach over a few ten or hundred thousand. By leaving them on your hard drives, you are slowly creating new knowledge tombs, with little prospects to be re-opened again.

But how can we re-evaluate knowledge from knowledge tombs? Evaluation of knowledge in the form of rules was mainly a manual job, usually limited to domain experts included in the initial study or research. Human factors, like limited availability, subjectivity, mood, etc., often influenced the evaluation and increased the possibility of missed opportunities.

Growth of internet usage and content in the last 10 years (according to some sources is world average from 400% on [1, 2]) provides us with large amounts of free, unstructured text information, which could be used to extend our data mining capabilities and to collect specialist knowledge from different more or less reliable sources.

And internet sources are the answer - using the web based text mined knowledge to verify knowledge, gathered in the knowledge tombs is approach we will present in this paper.

2 Knowledge Tombs and Forms of Rules for Re-evaluation

At the beginning, let us define the knowledge tombs.

Researchers and users of the artificial intelligence approaches, usually produce different sorts of knowledge, extracted from various data sources. After a quite intensive period of time, when all the evaluations and usage of the knowledge is done, that knowledge is stored in some sort of electronic form and with some ‘luck’ never to be used again.

Luckily, the times change and with the advancement of technology, expert knowledge is becoming more and more easily accessible through the information highway – the internet. In this paper we are going to address the most common form of internet knowledge, the free, natural language texts and present ways to mine it for knowledge, which will be used for re-evaluation.

Which knowledge tombs are we going to open? Typically, all white box, ‘rule producing’ machine learning and data mining approaches are the ones that produce knowledge in a form, appropriate for re-evaluation with our method.

In the following subsections, we are going to present the most typical representatives of the symbolic and ensemble approaches and their knowledge representations. The latter is crucial for us, because we need to know exactly what type of knowledge can be found in knowledge tombs.

2.1 Symbolic Machine Learning Approaches

Symbolic machine learning approaches try to capture knowledge in as symbolic a form as possible to provide a very natural and intuitive way of interpretation. Typically, decision trees, association rules, rough sets, and ensemble approaches are used for knowledge extraction. Ensemble methods usually perform better in comparison with single methods regarding classification accuracy, but they produce larger amounts of rules, which makes them the target group for our knowledge re-evaluation approach.

2.1.1 Decision Trees

Decision trees[3] are one of the most typical symbolic machine learning approaches, which have been present on the machine learning scene since the mid-1980s, when Quinlan presented his ID3 algorithm.[4] Decision trees take the form of a hierarchically-bound set of decision rules that have a common node, called a root. The rest of the tree consists of attribute (tests) nodes and decision (leaf) nodes labelled with a class or decision. Each path from a root node to any decision node represents a rule in a decision tree. Because of the very simple representation of accumulated knowledge they also give us the explanation of the decision that is essential in medical applications.

Top-down decision tree induction is a commonly used method. Building starts with an empty tree. Afterwards, a 'divide and conquer' algorithm is applied to the entire training set, where the most appropriate attribute is selected. Selection is based on a purity measure that determines the quality of the attribute split and represents a vital part of the method's background knowledge. A typical purity measure is some sort of derivative of an entropy function. Another very useful advantage of decision trees is the fact that they do not use all available attributes from the training set, but only those that are necessary for building a tree. Reducing the number of attributes (also called horizontal reduction) has very valuable consequences, since it provides information about which attributes are sufficient for a description of the problem and which are redundant.

The knowledge accumulated in the decision tree is represented in the form of a tree of rules, which can be easily transformed into a set of rules and also into a set of used attributes.

What we are interested in, are the rules, which can be easily obtained from the decision tree – each path from the root of the tree to a decision node, gives us one rule.

2.1.2 Association Rules

The association rule approach searches for frequently occurring and, therefore, interesting relationships and correlation relationships among attributes in a large set of data items.[5] Association rules show attribute-value conditions that occur together frequently in a given dataset. Association rules provide information in the form of 'if-then' statements. The rules are computed from the data and are of a probabilistic nature. The 'if' part of the rule is called the antecedent, while the 'then' part is called the consequent. The antecedent and consequent are sets of items that may not have any items in common. Each association rule also has two numbers that express the degree of uncertainty about the rule.

The first number is called the support and is the number of transactions that include all items in the antecedent and consequent parts of the rule. The second number is called the confidence of the rule and is the ratio between the number of transactions that include all items in the consequent, as well as the antecedent and the number of transactions that include all items in the antecedent.

Extracted knowledge of association rules is presented in the form of rules.[5] The difference between this method and the other presented approaches is in the consequent, where decision trees and rough sets use only one consequent attribute for all rules they produce on one data set.

2.1.3 Rough Sets

Rough sets are a symbolic machine learning approach based on classic set theory, which were introduced by Pawlak et al.[6] in 1995. They are often compared with other techniques, especially to statistical analysis and other machine learning methods. It is claimed that rough sets perform better on small data sets and on sets where data distribution significantly differs from uniform distribution.[6] In many cases, detailed data is not required to make decisions as approximate or rough data would be sufficient. Rough sets use reducts, a sufficient subset of attributes, to generate a rule set covering objects from the training set. Reducts are sets of attributes that are actually a subset of the entire set of attributes available in the training set. The rules that are obtained from the rough set approach can be either certain or uncertain. Certain rules are used in cases where there is background in consistent training objects. In contrast, uncertain rules are produced in cases where the training objects are inconsistent with each other. The latter situation usually presents a big hindrance for other machine learning approaches.

The main application areas of the rough set approach are attribute reduction, rule generation, classification and prediction.[6] Both rules and the set of attributes (reduct) are explicitly expressed, so there is no additional effort needed to extract the rules.

2.1.4 Ensemble Methods

Hybrid approaches in machine learning rest on the assumption that only the synergistic combination of different models can unleash their full power. The intuitive concept of ensemble approaches is that no single classifier can claim to be uniformly superior to any other, and that the integration of several single approaches will enhance the performance of the final classification.

To overcome some of disadvantages and limitations of a single method, it is sometimes enough to use different models of the same machine learning approach; e.g. using many different decision trees, neural networks or rough sets for the same training set.[7] In other cases, the approach relies on combining different machine learning or general problem solving methods. Typically, any machine learning methods can be combined in the form of an ensemble, but if symbolic machine learning methods are used, the condition about the possibilities of interpretation and explanation are satisfied. The two most popular ensemble techniques are called bagging[7] and boosting.[8] In simple terms, multiple classifiers are being combined into a voting body that is capable of making decisions with higher accuracy than any single classifier included in the voting body.

Bagging uses random selection of training objects with replacement from the original data set, to form a subset of objects, used for induction.[7] If random selection is used together with replacement, there is a possibility that in the training subset some training objects from original data set occur more than once and some do not occur at all. The drawback of bagging is its random selection, where luck is relied upon to select an appropriate subset of training objects. By counting the votes, the class with the most votes can be assigned to the unseen object. Bagging always improves classification accuracy of any included single classifier.

Boosting uses the idea behind bagging and extends it further. Boosting tries to overcome the problem of random selection in bagging by assigning weights to the training objects and looking back to see how successful the previously induced classifier was.[8] This makes this approach incremental. If a training object was classified incorrectly, its weight was increased, and if it was classified correctly, its weight was decreased. Weights play the main role in selecting training objects for the next iteration of classifier induction, since the focus is on training objects that have higher weights and are, therefore, harder to classify correctly. The final classifier consists of earlier classifiers that are better on ‘easier’ training objects with lower weights and latter classifiers, which are more specialized in classifying ‘harder’ cases with higher weights.

Since both bagging and boosting can be used on wide variety of basic machine learning methods, we are going to limit ourselves only to symbolic white box approaches, where the rules are at hand or can be easily extracted from each member of the ensemble.

2.2 Web Based Text Mining

Text mining is a relatively new research area with first attempts going back approximately 15 years. In that time some authors tackled the problem of extracting company names from a free text. The amount of electronic texts and potential knowledge grew over all expectation since the beginning of internet. It is also known that most of the knowledge found on the internet is unrelated and unstructured. But even on that field a big step forward was made in the recent years – systems for textual information extraction became a key tool for news aggregators, internet people searchers, thesaurus generators, indexing tools, identification of proteins, etc. That kind of systems are not ‘smarter’ than people, but have two great advantages: they are consistent and much faster than people.

Most of the systems work as unilingual version, but the real challenge represent multilingual and language invariant approaches. Most multilingual approaches have language recognition preprocessors. There are some parallels between mining different languages and specific knowledge domains (using the same language) like medicine, biomedicine, and military.

We believe, that the using the knowledge from different internet sources can take some burden off the shoulders of domain experts and enable faster knowledge acquisition.

In general, we can describe the text mining with the workflow in Fig. 1. Since text mining is a natural language processing, it can be accessed from the same levels as the natural language analysis [9]:

- **Prosody** analyses rhythm and intonation. Difficult to formalize, important for poetry, religious chants, children wordplay and babbling of infants.
- **Phonology** examines sounds that are combined to form language. Important for speech recognition and generation.
- **Morphology** examines word components (morphemes) including rules for word formation (for example: prefixes and suffixes which modify word meaning). Morphology determines the role of a word in a sentence by its tense, number and part-of-speech (POS).
- **Syntax** analysis studies the rules that are required for the forming of valid sentences.
- **Semantics** studies the meaning of words and sentences and the means of conveying the meaning.
- **Pragmatics** studies ways of language use and the effects the language has on the listeners.

In our work we will be focusing on the morphology, syntax and semantic levels.

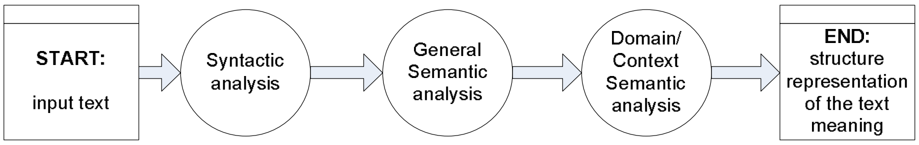

START:
input text
Syntactic analysis
General Semantic analysis
Domain/ Context Semantic analysis
END: structure representation of the text meaning

Fig. 1. Natural language analysis process

When considering means to acquire knowledge from natural language sources the analysis is a three step process: **syntactic analysis**, **meaning analysis** (semantic interpretation; generally in two phases) and the **forming of the final structure** that represents the meaning of the text.

In the following section we will continue with a somewhat simplified example of knowledge extraction from natural language texts. The example will be based on natural language passages and how to extract formalised data from them. The two main steps in the process are the following:

1. Acquisition of the natural language resources and pre-processing.
2. Knowledge extraction and formalization.

2.2.1 Acquisition of the Natural Language Resources and Pre-processing

The acquisition process is the process in which we define the source of data and means to acquire it. It can be as simple as gathering the documents to a central storage point or it can be the implementation of a web crawler that will investigate the target web sites and transfer the data to the central storage point.

The essential steps in the preprocessing are two. The first is the transformation of the documents to plain text and the second is the tokenization. While the former is an entirely technical issue that can be successfully solved without significant effort, the latter requires much thorough approach and is far from trivial. The tokenisation is essential for the passage and sentence level semantic analysis. However some semantic information is required for the successful resolution of the meaning of punctuation (for instance whether a period ends a sentence or just an abbreviation). The simple implementation where the period is assumed to end a sentence proved to achieve a successful tokenization rate of just under 90% on the Penn Treebank corpora. Unfortunately that is not enough, and a higher success rate is required because of an error in tokenisation, which is usually magnified by several orders in the analysis phase.

2.2.2 Knowledge Extraction and Formalisation

The first step in knowledge extraction is the part-of-speech (POS) analysis. It can be performed with the use of existing POS taggers, although it is highly language dependant. In the example we are presenting we will assume that the source documents have been gathered, transformed to plaintext and tokenised to individual sentences. The sentences to be used for semantic analysis can be classified by statistical metrics. Let us assume that a sentence has been identified. The sentence is stated as:

"Eating as little as a quarter of an ounce of chocolate each day may lower your risk of experiencing heart attack or stroke!".

The POS analysis provides the tags listed in Table 1.

Table 1. POS tags of a news sentence

Word	Tag	Word	Tag	Word	Tag
Eating	VBG	*as*	RB	*little*	JJ
as	IN	*a*	DT	*quarter*	NN
of	IN	*an*	DT	*ounce*	NN
of	IN	*chocolate*	NN	*each*	DT
day	NN	*may*	MD	*lower*	VB
your	PRP$	*risk*	NN	*of*	IN
experiencing	VBG	*a*	DT	*heart*	NN
attack	NN	*or*	CC	*stroke*	VB
Abbreviations: IN - Preposition or subordinating conjunction, JJ - Adjective, MD - Modal, NN - Noun, singular or mass, PRP$ - Possessive pronoun, RB - Adverb, VB - Verb, base form, VBG - Verb, gerund or present participle					

Semantic interpretation uses both the knowledge about word meanings (within the domain) and linguistic structure. The analysis produces a representation of the meaning of the sentences in an internal format. This can be visualised as in Fig. 2.

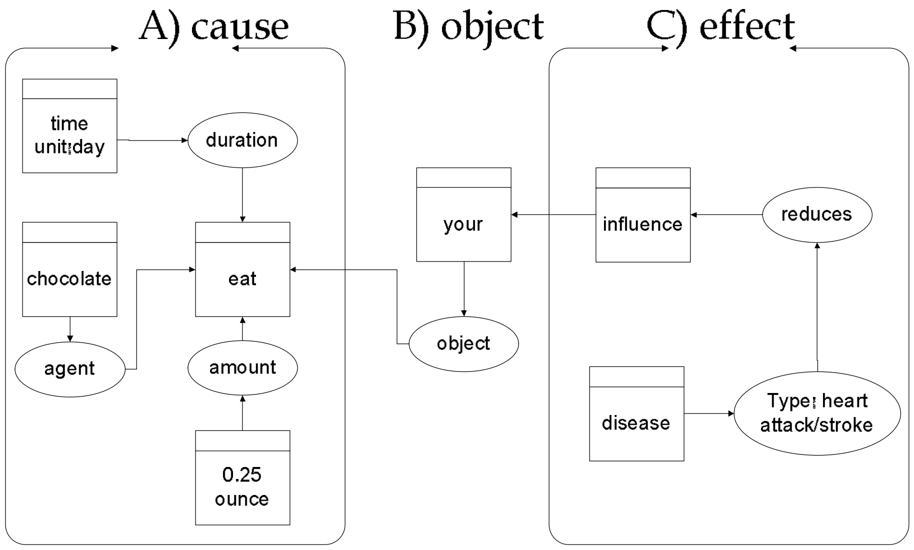


Fig. 2. Internal representation of the meaning of the sentence

The process of the analysis is described in the following. The sentence is separated into two distinct categories: cause (IF) and effect (THEN). Both are associated with the object. In the figure the application used knowledge that *ounce* is a unit of amount, *day* is a unit of time and that a person normally eats chocolate not the other way around. So combining this knowledge produced the resulting representation of knowledge in the sentence. The agent (the one that influences) is *chocolate*, the object (the recipient of the action) is the word *your* and the action (agent to object) is *eating*. Combining that to *eat* is associated with the domain concept of amount and that *ounce* is a unit of amount the application can effectively reason that the meaning of the cause part (Fig. 2 segment A) of the sentence is: *object that eats a 0.25 ounce of chocolate in a period of one day*. The effect side (Fig. 2 segment C) has the meaning of: *the object experiences the influence of reduced possibility of a disease of type heart attack/stroke*. This internal representation is then generalized with the addition of known concepts. The object *yours* is a possessive pronoun and is therefore mapped to a person which is marked as "*patient*„ in the domain.

The amount of *quarter of an ounce* is mapped to the primary unit for amount in the domain, (*grams*) with the use of a conversion factor. So ¼ of an ounce becomes 7.08738078 grams. The resulting semantic net (Fig. 2) with the additional information is the final interpretation of the domain specific world knowledge learned from this sentence.

The form shown in Fig. 2 is the form that can be used for the formalisation of knowledge. The formalisation is the process of storing the knowledge in a formal, machine readable form that can be used as the need arises by various types of intelligent systems. A common formalisation approach is the transformation to rules. For the example we have been following a rule would be in the following form:

```
RULE chocolate consumption influence
  IF typeof (object) IS patient
    AND typeof (action) IS eat
    AND action::target IS chocolate
    AND  quantityof (action) IS 7g
    AND  timespan (action) IS 24h
  THEN typeof(consequence) IS influence
    AND      consequence::target IS disease
    AND  typeof(disease) IS heart attack/stroke
    AND relationship (consequence,
consequence::target) IS reduced risk
```

This is the final formalization of acquired knowledge. In this form the knowledge is fully machine readable, providing there are inferring rules that define how to evaluate the value entities (*typeof*, *quantityof*,…). This format can be stored and used as need arises.

What about the rules, we want to re-evaluate? Even they must be checked and put into context. To do that, we have to process the descriptions of the underlying database, which usually contain the descriptions of the problem domain and the attributes used in the database and (consequently) in the rules produced by the machine learning method. This process is a simpler version of the process, described above, since the rules are already in the formal, machine readable structured form.

2.2.3 Re-evaluation of the Rules and Search for New Knowledge

Finally we come to the part, where we can start to compare the rules, which are in the same formal form, but come from different data sources.

Our goal is to find support for machine learning rules, which we brought from our knowledge tomb in the text we mined from our natural language resource and is now also in a form, suitable for comparison. We will be looking for the highest match between the cause(s) and effect(s) between individual representatives from both sets of rules.

After a series of automatic procedures, there is again time, to involve a human operator, a domain expert, who will examine machine learning rules with the top ranked support, found on the web. His task is to extract new and potentially new knowledge by examining the machine learning rules and by standing rules with text sources, from which they were derived.

Explanations in the natural language are there to encourage thinking from different perspectives and hopefully provide a reasonable explanation and help at revealing new or unconscious knowledge. It is up to expert's expertise to recognize, expose and reason about the possible new knowledge, but now with higher level of support, than years ago, when the knowledge tomb was created.

3 Discussion and Conclusions

Knowledge mining from databases of solved cases with machine learning methods is nothing new in community involved with artificial intelligence. Used approaches are very different, but most of them have a common point –machine learning approaches and their outcomes. Results of such approaches are in general sets of rules with some additional information, which are capable to generalize knowledge from database of solved cases or to determine associations between attributes in the data base.

It was usual practice that the generated set of rules was checked by a domain expert (for example a medical doctor), which used his knowledge and experience to evaluate rules as senseless or sense, and the latter to known or potentially new. The last type of knowledge is the most interesting for us, because by discovering new knowledge we can find solutions to unsolved cases or find alternative solutions for solved problems. It is our experience that new knowledge is very hard to find so we decided to find a way to automatically support a part of knowledge evaluation.

Re-evaluation of ML rules is a process which increases the potential of already generated, but disregarded rules and hopefully triggers the 'Aha!' effect which accompanies the transformation of a rule in to a new knowledge.

The described re-evaluation should not become only one-time event, but should become a process which takes part on a regular basis.

Faster and more consistent knowledge verification reduces the need for manual domain expert work and shortens the cycle *'searching for potential knowledge – verifying potential knowledge – using new knowledge'*.

Of course there are some concerns, which we are aware of and present a pit fall for the re-evaluation process. With the increased amount of information, available on the internet, there is also a vast number of sources we cannot trust and because of that, the involvement of a domain expert in the final stage of knowledge re-evaluation remains a necessity.

References

1. Internet Growth Statistics - Today's road to e-Commerce and Global Trade (2010), `http://www.internetworldstats.com/emarketing.htm`
2. Malik, O.: Big Growth for the Internet Ahead, Cisco Says (2010), `http://gigaom.com/2008/06/16/big-growth-for-internet-to-continue-cisco-predicts/`
3. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann, San Francisco (1993)
4. Quinlan, J.R.: Induction of decision trees. Machine Learning, 81–106 (1986)
5. Piatetsky-Shapiro, G.: Discovery, analysis, and presentation of strong rules. In: Piatetsky-Shapiro, G., Frawley, W.J. (eds.) Knowledge Discovery in Databases, pp. 229–248. AAAI/MIT Press, Cambridge (1991)
6. Pawlak, Z., Grzymala-Busse, J., Slowinski, R., et al.: Rough sets. Communications of the ACM 38, 89–95 (1995)
7. Breiman, L.: Bagging predictors. Machine Learning 24, 123–140 (1996)
8. Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. In: Machine Learning: Proceedings of the Thirteenth International Conference, pp. 148–156. Morgan Kauffman, San Francisco (1996)
9. Luger, G.F.: Artificial intelligence, Structure and Strategies for Complex Problem Solving, 5th edn. Pearson Education Limited, USA (2005)

Faceoff: Surrogate vs. Natural Keys*

Slavica Aleksic[1], Milan Celikovic[1], Sebastian Link[2], Ivan Lukovic[1], and Pavle Mogin[2],**

[1] University of Novi Sad, Serbia
[2] Victoria University of Wellington, New Zealand
pmogin@ecs.vuw.ac.nz

Abstract. The natural and surrogate key architectures are two competing approaches to specifying keys in relational databases. We analyze the performance of these approaches with respect to the update complexity and the query data and expression complexity. Our results provide new insights into the advantages and disadvantages of both approaches.

1 Introduction

Keys are a class of database constraints that is fundamental to any data model and of a great importance to data modeling and most data processing tasks. In practice, there are two competing approaches to specifying keys: the natural and the surrogate key approach. A natural key is an attribute subset of the underlying relation scheme. The existence of a natural key is known to database users. The major perceived disadvantage of natural keys is their susceptibility to changes in both value and structure. The use of natural primary keys is often discouraged in industry. Nevertheless, natural keys ought to be specified whenever they represent business rules of the underlying application domain. In the surrogate key architecture, each relation scheme has a surrogate attribute as its primary key, and surrogate primary keys are propagated to other relation schemas as foreign keys. A surrogate key is a single attribute whose values are (i) numeric, (ii) system generated, and (iii) used to identify tuples of a relation. Its existence and values are invisible to users. A perceived advantage of using surrogate primary keys is their immutability, a consequence of their separation from business logic. Codd proposed the use of the surrogate key database architecture, but did not consider its impact on the database performance [1]. During 2009 the authors conducted a survey of database design and implementation practices in industry, where a prevailing number of respondents claimed to use a surrogate key architecture [3]. To the authors best knowledge no comprehensive comparison of both key architectures is available in the literature. It is therefore the main objective of this article to estimate the consequences of either implementation choice on the complexity of updates and queries.

* This research was supported by Victoria University of Wellington, Wellington, New Zealand Research Grant 85575.

** Corresponding author.

2 Performance Estimates

As an illustrative showcase we investigate the performance differences of the natural and surrogate key architectures that both correspond to the chain of ER relationship types [4] in Figure 1. For this purpose, the ER diagram is mapped to two relational database schemes S and S', respectively [3]. The schema S is based on natural, while the schema S' is based on surrogate keys. For $i = 1, \ldots, 4$, we denote the natural primary key of E_i by nk_i. To guarantee the uniqueness of tuples, the relationship relation schemes R'_1, R'_2 and R'_3 in S' require an additional candidate key, composed of propagated surrogates.

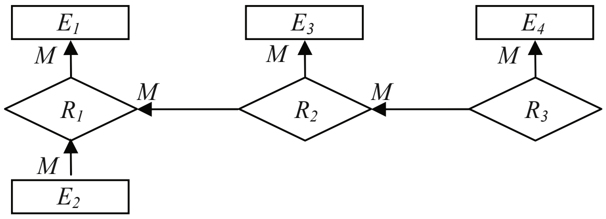


Fig. 1. A Higher-Order Entity-Relationship Diagram

Update complexity. We consider the effects of inserts, deletes, and modifications made to entity and relationship relations separately, since updates of relationship relations have a considerably greater impact on the overall database performance. Our analysis [3] shows that inserts and deletions in relations over the entity type E_i are performed more efficiently. Contrary to that, modifications of natural key values should be cascaded through relationship relations and that may lead to a poor performance. A natural key modification in a relation over entity type E'_i does not propagate. Updates of relations over R_i are accomplished by issuing solely SQL INSERT, DELETE, and UPDATE commands. Updates of surrogate key relationship relations require the retrieval of appropriate surrogate key values before issuing SQL update commands. To make the last claim more obvious, we consider updates of relations over R'_3. Before issuing an update command, a program controlling updates, has to execute the following two queries:

SELECT sk_6 FROM E'_1 NATURAL JOIN E'_2 NATURAL JOIN R'_1 NATURAL JOIN E'_3 NATURAL JOIN R'_2 WHERE $nk_1 = a_1$ AND $nk_2 = a_2$ AND $nk_3 = a_3$;

and

SELECT sk_4 FROM E'_4 WHERE $nk_4 = a_4$;

where a_i $(i = 1, \ldots, 4)$ are natural key values. The two queries above are needed since surrogates are invisible to users and they have to define their updates in terms of natural keys. The two queries have to be issued irrespective of the kind of the update command. Due to the need to retrieve surrogate key values using queries and to update an additional B-tree, it may be predicted that updates of relationship relations in surrogate key databases take longer to execute.

Query complexity. In this section, we analyze the following queries:

Q1 Retrieve values of a non-unique attribute A_3 of R_3 for a conditional expression on the value k_1 of the natural key nk_1.
Q2 Retrieve values of the natural key nk_1 for a conjunctive conditional expression on the non-unique attribute B from the entity relation E_1 and A_i from relationship relations R_j, for $j = 1, 2, 3$.

We assume that the query data complexity results primarily from the number of joins, and secondarily from the number of disk blocks. A detailed discussion of performance estimates of a larger number of queries may be found in [3]. Following the natural key architecture, query *Q1* maps into the SQL expression:

$$\texttt{SELECT } A_3 \texttt{ FROM } R_3 \texttt{ WHERE } nk_1 = k_1;$$

and following the surrogate key architecture into

$$\texttt{SELECT } A_3 \texttt{ FROM } R'_3 \texttt{ NATURAL JOIN } R'_2 \texttt{ NATURAL JOIN } R'_1 \texttt{ NATURAL JOIN } E'_1 \texttt{ WHERE } nk_1 = k_1; \quad .$$

The value of the natural key nk_1 is available in the natural key database from R_3 directly, and only from the entity relation E'_1 in the surrogate key database. Accordingly, the natural key database has a clear performance advantage over the surrogate database.

The query *Q2* maps into the SQL expressions:

$$\texttt{SELECT } nk_1 \texttt{ FROM } R_3 \texttt{ NATURAL JOIN } R_2 \texttt{ NATURAL JOIN } R_1 \texttt{ NATURAL JOIN } E_1 \texttt{ WHERE } B = b \texttt{ AND } A_1 = a_1 \texttt{ AND } A_2 = a_2 \texttt{ AND } A_3 = a_3;$$

for the natural key architecture, and into

$$\texttt{SELECT } nk_1 \texttt{ FROM } R'_3 \texttt{ NATURAL JOIN } R'_2 \texttt{ NATURAL JOIN } R'_1 \texttt{ NATURAL JOIN } E'_1 \texttt{ WHERE } B = b \texttt{ AND } A_1 = a_1 \texttt{ AND } A_2 = a_2 \texttt{ AND } A_3 = a_3;$$

for the surrogate key architecture. Both SQL expressions require the same number of joins. Here, we estimate that a smaller number of blocks favors the performance with respect to the surrogate key architecture.

3 Experiments

To validate our previous findings we have implemented a natural key and a surrogate key database on an ACPIx86 - based PC@1.6GHz with 2 GB RAM. The operating system was the 32-bit Windows Vista Home Basic. The DBMS was Oracle 10g. The relationship relations were populated in the following way: R_1 and R'_1 with 10^4 tuples, R_2 and R'_2 with 10^5, and R_3 and R'_3 with 10^6 tuples. Performance testing was performed using Oracle SQL Developer.

Table 1. Average database update and query times in seconds

Update	Natural Key	Surrogate Key	Surrogate / Natural
Inserts	0.0753	0.4703	6.246
Deletes	0.0697	0.5116	7.636
Modification of nk	4.6412	0.0072	0.002
Query Q1	0.0493	0.2510	5.091
Query Q2	0.1294	0.0674	0.521

The first part of Table 1 contains average times of tuple inserts and deletes in relations over R_3 and R'_3, as well as the average time of the modification of a natural key nk. The insert and delete average times closely follow our earlier predictions. The surrogate key database significantly outperforms the natural key database, when a modification of a natural key is considered. The last two rows of Table 1 display the average times of executing queries defined in Section 2 and agree with query performance estimates presented. There is no clear evidence that any of the architectures noticeably outperforms the other.

4 Conclusion and Future Work

In this paper, we have compared the natural and surrogate key relational database architectures. Utilizing mathematical reasoning and a performance analysis in the Oracle 10g DBMS we have obtained the following major insights:

– The natural key architecture performs considerably better when updates to relationship relations are considered.
– The modification of natural key values is a great disadvantage of the natural key architecture.
– If deep hierarchies of higher-order relationship relations are present, then the query performance of neither of the architectures is superior to the other.

In future work we plan to conduct experiments with DBMSs different from Oracle, to extend classes of queries considered, to analyze the effect of the natural key propagation in surrogate key relationship relation schemes, and to consider different data models including XML [2].

References

1. Codd, E.F.: Extending the database relational model to capture more meaning. ACM Trans. Database Syst. 4(4), 397–434 (1979)
2. Hartmann, S., Link, S.: Efficient reasoning about a robust XML key fragment. ACM Trans. Database Syst. 34(2) (2009)
3. Link, S., Lukovic, I., Mogin, P.: Performance evaluation of natural key and surrogate key database architectures. Technical Report ECSTR10-06, Victoria University of Wellington, New Zealand (2010)
4. Thalheim, B.: Entity-Relationship modeling - foundations of database technology. Springer, Heidelberg (2000)

An Optimal Relationship-Based Partitioning of Large Datasets

Darko Capko, Aleksandar Erdeljan, Miroslav Popovic, and Goran Svenda

Faculty of Technical Sciences, Trg Dositeja Obradovica 6, 21000 Novi Sad, Serbia
dcapko@uns.ac.rs, erdeljan@uns.ac.rs, miroslav.popovic@rt-rk.com,
svenda@uns.ac.rs

Abstract. Modern adaptive applications utilize multiprocessor systems for efficient processing of large datasets where initial and dynamic partitioning of large datasets is necessary to obtain an optimal load balancing among processors. We applied evolutionary algorithms (Genetic Algorithm and Particle Swarm Optimization) for initial partitioning, and diffusion (DR) and cut-and-paste (CP) algorithms for dynamic partitioning. Modified versions of DR and CP algorithms are developed to improve dynamic partitioning running in NUMA multiprocessor systems. The proposed algorithms were applied on datasets describing large electricity power distribution systems and experimental results prove reductions of processor load imbalance and performance improvements.

Keywords: Graph partitioning, diffusion algorithms, NUMA.

1 Introduction

Today many applications use large datasets and utilize multiprocessor systems in order to minimize overall response time when data is continually changed. In such parallel computing environments, data parallelism is used as a form of parallelization of computing across multiple processors. In our research we assume that large datasets can be divided into smaller data subsets which will be used by parallel tasks as independent data partitions. The aim is to optimally divide data in order to minimize data relations across these subsets. Also, by making partitions of the similar size the computation load of every processor in the multiprocessor system might be balanced. In order to achieve this balance a graph is created out of the datasets as vertices and their relations as edges, and the optimization problem is set as a problem of graph partitioning. In this paper we exploit NUMA (Non-Uniform Memory Architecture) where the memory access time decreases when datasets inside a partition are positioned into the same NUMA node.

In a typical parallel computing environment we are considering here, tasks are producing results out of a group of data subsets including input values $U = \{U_r, U_o\}$, where U_r denotes values that have influence on relations among datasets, while the rest of the input values U_o affect only calculation results. When U_r is changed some relations inside the subsets are affected, resulting in a possible slower task execution caused by the references to other datasets out of its local memory. Relocating data between NUMA nodes could bring better performances, although it might introduce an unbalanced system. Therefore, an optimal partitioning of the datasets is required and it is applied in two complementary ways: 1) initially - before starting the system, and 2) dynamically - while the system works (on-line).

2 Problem Definition

In our discussion we study a system that continually processes large datasets in terms of periodical or trigger based executions of various functions. Let Q denotes large datasets and e elements it contains. If two elements e_i and e_j are in relation $N(e_i, e_j)$ we assume that the processing function ξ will use them together (only relations meaningful to ξ are considered). This also means that the elements are *connected* and they are called *neighbors*. The relation between neighbors that depends on an input value $u \in U_r$ could be temporarily inactive and we call it *potential connection* $Pot(e_i, e_j, u)$. In other words, when a potential connection is activated two elements become neighbors

$$(\forall e_i, e_j \in Q)Pot(e_i, e_j, u = active) \Leftrightarrow N(e_i, e_j) \tag{1}$$

or vice versa. The set of mutually connected elements is called *region* R

$$(\forall e_i \in R)N(e_i, e_j) \Leftrightarrow e_j \in R_k, \quad k \in \{1, 2, \ldots, n\} \tag{2}$$

which is the smallest data unit that can be processed by ξ. All regions create a *calculation domain* D ($D = R_1 \cup R_2 \cup \ldots \cup R_n$) and function $Y_i = \xi(R_i)$ is applied to each region R_i to produce output result set $Y = Y_1 \cup Y_2 \cup \ldots \cup Y_n$.

In a multiprocessor environment function ξ can be applied to individual regions in parallel, and if the number of regions is bigger than the number of processors, regions are grouped into m partitions P_i and distributed to m processors. The regions could change over time because the connections among the elements depend on the input datasets (1). When a potential connection between the elements from the different regions is activated, the two regions have to be merged, and vice versa, it is expected that deactivated potential connection could cause a splitting of the region. We use graphs to present Q datasets and leverage k-way graph partitioning that has been successfully applied to many areas. The resulting domain D is described as a weighted undirected graph, G = (V,E) made of vertices (V) and edges (E). A vertex represents region R_i with weight w_i as time needed to execute $\xi(R_i)$. Graph edges represent potential connections between elements from the different regions. Two regions p, q may have many potential connections $Pot_{p,q}$ and we present them using a single edge the weight of which is equal to the total number of such connections.

Further, in order to define optimization criteria, for a partition k we need to define partition weight $W_k = \sum_{j \in P_k} w_j$ as the sum of contained regions weights, and function ϕ_k as:

$$\phi_k = \sum_{p,q \in P_k} Pot(p, q) \tag{3}$$

where regions p and q are in P_k. At the beginning it is necessary to group the regions into a defined number of partitions (n_p), so that these partitions are approximately of same weights, but never greater than the maximal partition weight $M = \frac{1+\epsilon}{n_p} \sum W_k$, where W_k is partition weight and $\epsilon \in [0, (n_p - 1)/n_p]$ is the tolerance. The optimization criterion should obtain the maximum connection inside a partition:

$$F = \max \sum_k \phi_k \tag{4}$$

where function ϕ_k is given by (3), and all partition weights are constrained $W_k \leq M, \forall k \in \{1, 2, \ldots, n_p\}$.

3 Algorithms and Experimental Results

For initial partitioning we applied evolutionary computation techniques *Genetic Algorithm* (GA) [1] and *Particle Swarm Optimization* (PSO) [2] to find out the best approximate solutions for the optimization problem. Then some potential connections were activated making an unbalanced system. Four algorithms for dynamic partitioning: *diffusion* (DR) [4] and *cut-and-paste* algorithm (CP) [3] with their modifications were applied and compared.

Modified Diffusion Repartitioning (MDR) algorithm considers two hierarchical levels, and it is suitable for NUMA. At the first level DR is applied to balance the sum of partition weights for all NUMA nodes, while at the second level the balance among partitions associated with a NUMA node is made. After NUMA nodes are balanced, DR algorithm is applied to rearrange partitions in each node.

Modified Cut-and-Paste Repartitioning (MCP) algorithm also uses two level partitioning. At the first level the CP algorithm is applied only to those partitions that are placed on a different NUMA node, while at the second level the CP algorithm is applied only to regions that are associated with the NUMA node.

We tested the initial and the dynamic partitioning algorithms on large datasets used in power distribution utilities, for CIM connectivity/topology models [5]. Results of tests presented in Figure 1 were made on two models: *it206* (with 1787939 elements transformed into a graph with 206 vertices used for tests T1 and T2) and *bg5x* (5980390 elements/315 vertices model used for tests T3 and T4). In tests T1 and T3 new connections were made between regions that belong to different NUMA nodes, while tests T2 and T4 considered connections in the same NUMA node. Experiments were carried out on the NUMA platform with 2 nodes with 4 cores per node (CPU AMD Opteron 2.4GHz, 8GB RAM per core).

MDR obtains better results than the other algorithms because of a smaller total migration size and good results for function F, although it executes slower than other algorithms. On the other hand, MCP is the fastest algorithm that obtains very good results in terms of data migration, however it is not as good for function F optimization as MCP is. Considering that connections between partitions are potential connections, this criterion is less significant to us.

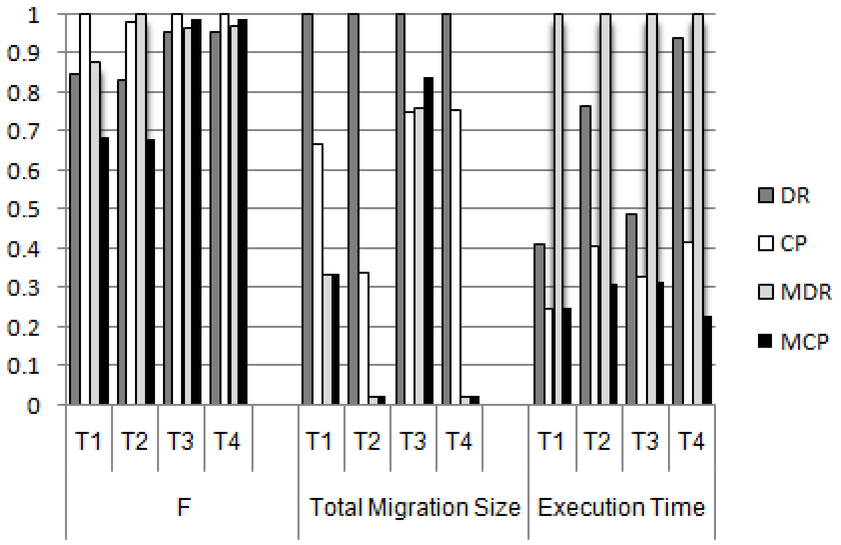


Fig. 1. Normalized optimization criteria F, total migration size and execution time obtained by DR, CP, MDR and MCP algorithms on tested graphs

4 Conclusion

GA and PSO are both successfully applied for initial partitioning and DR and CP algorithms for dynamic partitioning as well. We developed modified algorithms MDR and MCP to exploit NUMA for dynamic partitioning and they have shown better performances than original algorithms. We recommend MCP for dynamic partitioning in real-time systems as the fastest algorithm.

References

[1] Bui, T.N., Moon, B.R.: Genetic Algorithm and Graph Partitioning. IEEE Transaction of Computers 45(7), 841–855 (1996)
[2] Laskari, E., Parsopoulos, K., Vrahatis, M.: Particle swarm optimization for integer programming. In: Proceedings of the IEEE Congress on Evolutionary Computation, Honolulu, Hawaii USA, vol. 2, pp. 1582–1587 (2002)
[3] Schloegel, K., Karypis, G., Kumar, V.: Multilevel diffusion schemes for repartitioning of adaptive meshes. Journal of Parallel and Distributed Computing 47(2), 109–124 (1997)
[4] Schloegel, K., Karypis, G., Kumar, V.: Wavefront diffusion and LMSR: Algorithms for dynamic repartitioning of adaptive meshes, Technical Report TR 98-034, Dept. of Computer Science and Engineering, University of Minnesota (1998)
[5] IEC 61970 Energy management system application program interface (EMS-API) - Part 301: Common Information Model (CIM) Base", IEC, Edition 2.0 (2007)

Design and Semantics of a Query Language for Multidimensional Data

Ingo Claßen

Hochschule für Technik und Wirtschaft Berlin
ingo.classen@htw-berlin.de

Abstract. This paper introduces a new language for analysis of multidimensional data. Its design is discussed and a reference to a formal semantics is given.

1 Introduction

Multidimensional structures play an important role for the analysis of information within data warehouse systems (see, e.g., [MZ09, KR02]). Special language concepts have been developed to support this kind of analysis, see, e.g., [GL97], [PRP02], [CT97], [LW96] and a concrete language called MDX (multidimensional expressions) [SC09, SHWC05] has been adopted by many vendors of OLAP tools. The latter seems to have no rigid definition of its semantics.

The aim of this paper is to present the design of a new language (called DimQL, dimensional query language) that relies on central ideas of MDX and that has a formal semantics. The main goal of the design was the ability to specify complex queries in an easy and natural way such that the language can be used on the conceptual level. The underlying semantics assures that specifications can be interpreted without being bound to a concrete implementation technology.

2 Language Design

This section explains the basic ideas behind the design of DimQL. We assume to have a multidimensional database of a video store containing dimensions like `Store`, `Product`, `Customer`, `Time`, attributes like `City`, `SGroup`, `MGroup`, `Band`, `Month`, `Year`, measures like `Return`, `Quantity` and members of attributes as given in the following tables.

Attribute	Members
`City`	`Boston`, `New York`
`SGroup`	`Crime`, `SciFi`, `Life`, `Technic`
`MGroup`	`Fiction`, `Non Fiction`

Attribute	Members
`Band`	`Junior`, `Adult`, `Senior`
`Month`	`2007Jan`, . . ., `2009Dec`
`Year`	`2007`, `2008`, `2009`

`Crime` and `SciFi` are subgroups of main group `Fiction` whereas `Life` and `Technic` are subgroups of `Non Fiction`. Attribute `Band` is meant to describe age bands.

2.1 Basic Queries

We start with the following DimQL query that delivers a two-dimensional report of aggregated values for all years and all main groups.

```
apply Rent(mg, y, Measures[Return]) over
  mg <- MGroup[*], y <- Year[*]
```

The two axes of the report are specified by `mg <- MGroup[*]` and `y <- Year[*]`. Expressions like `MGroup[*]` are called *member expressions* and in this example, we get the list of all main groups. `Rent(mg, y, Measures[Return])` is to be interpreted as a cell function that extracts return values from the underlying cube wrt. `mg` and `y` which are bound to main groups and years, respectively.

In the following example, a different cell function `Rent(mg, y, Band[Junior], Measures[Return])` is used.

```
apply Rent(mg, y, Band[Junior], Measures[Return]) over
  mg <- MGroup[*], y <- Year[*]
```

In this case, the cells of our report do not contain aggregated values for years and main groups but only aggregations for values that belong to age band `Junior`. The member expression `Band[Junior]` delivers the `Junior` member of attribute `Band`.

If we like to see returns wrt. years, main groups, and cities, three axes can be used.

```
apply Rent(mg, y, c, Measures[Return]) over
  mg <- MGroup[*], y <- Year[*], c <- City[*]
```

To show the resulting report, a three-dimensional representation is needed. A two-dimensional form of the same content can be achieved as follows.

```
apply Rent(mg, y, c, Measures[Return]) over
  mg <- MGroup[*],
  y, c <- Year[*].cj(City[*])
```

Here the cross-join member operator `cj` has been used and the left-hand side must now consist of two variables.

Up to now all axis members have been derived from instance sets of attributes. In the following example return and quantity of video rentals per year are to be shown.

```
apply Rent(m, y) over m <- Measures[Rent], y <- Year[*]
```

In the resulting report, measure names take the same role as main groups in our first DimQL query. Especially, there is a variable `m` for measure names. The member expression `Measures[Rent]` delivers all measures defined in cube `Rent`.

2.2 Member Expressions

To map complex requirements to queries, powerful mechanisms to form member expressions are needed. The idea behind DimQL to achieve this, is operator chaining. In general, a member expression is of the form

```
mspec.op1(...).op2(...) ... .opn(...)
```

where `mspec` is a member specification, i.e., a basic expression like `MGroup[*]` and `op(...)` is the application of a *member operator*. The whole expression can be regarded as a pipeline where each operator gets a set of members as input and delivers an output set of members.

Assume, we want to concentrate our attention on years where a minimum return of $1,000,000 has been made.

```
Year[*].filter(y -> Rent(y, Measures[Return]) >= 1000000)
```

This member expression delivers the wanted members by filtering the set of all year instances by predicate `Rent(y, Measures[Return]) >= 1000000`. In this expression variable `y` is bound to all years and for each binding the predicate is evaluated. Note that a cell function is applied within the predicate. Filtering `City` `Year` combinations can be done in the same way.

```
City[*].cj(Year[*]).filter(c,y -> Rent(c, y, Measures[Return]) >= 100000)
```

In this case the input to `filter` consists of `City` `Year` pairs.

2.3 Cell Function Expressions

The examples so far only have applied cell functions that directly correspond to measures of a given cube but DimQL is not confined to them. It provides means to construct cell functions, as can be seen in the following example where the report delivers some kind of return per quantity.

```
apply Rent(mg, y, Measures[Return])/Rent(mg, y, Measures[Quantity]) over
  mg <- MGroup[*], y <- Year[*]
```

Common operators like multiplication and division can be used to combine cell functions.

A second way for the construction of cell functions is by use of aggregation operators like `sum` and `avg`. In the following example, a report is produced that, for all age bands and all measures of cube `Rent`, calculates the average value of that measure wrt. subgroups `Crime` and `SciFi`.

```
apply SGroup[Crime, SciFi].avg(Rent(m, b)) over
  m <- Measures[Rent], b <- Band[*]
```

2.4 Calculated Members

In the following example, we find a cell function declaration `Total` that is defined by cell function expression `Rent(mg, Measures[Return])`. Moreover, there is a calculated member `Total` of type `Year`.

```
with
  Total(mg: MGroup) = Rent(mg, Measures[Return])
apply Rent(mg, y, Measures[Return]) over
  mg <- MGroup[*], y <- Year[*, Total]
```

The cells within the report can be calculated in the same way as explained in our first DimQL query, except for intersection points with calculated members. Intersection point `Total`, `Fiction`, e.g., leads to the application of `Total`(`Fiction`) that in turn leads to `Rent(Fiction, `*`Measures`*`[Return])` thus delivering the total over all years for subgroup `Fiction`.

3 Semantics

The semantics of DimQL queries are intended to be reports. Therefore, a formal treatment of this idea leads to the following function.

$$[\![\_]\!] : \mathit{DimQL} \hookrightarrow \mathit{Report}$$

Here, *DimQL* is the syntactical domain according to the starting nonterminal of the grammar and *Report* is the domain of reports. The definition of the semantics can found in [Cla10].

References

[Cla10] Claßen, I.: Towards a semantical foundation of DimQL. Hochschule für Technik und Wirtschaft, Berlin (2010), http://opus.kobv.de/htw/volltexte/2010/41/

[CT97] Cabibbo, L., Torlone, R.: Querying multidimensional databases. In: 6th International Workshop on Database Programming Languages, pp. 319–335 (1997)

[GL97] Gyssens, M., Lakshmanan, L.V.S.: A foundation for multi-dimensional databases. In: VLDB, pp. 106–115 (1997)

[KR02] Kimball, R., Ross, M.: The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling. John Wiley & Sons, Inc., New York (2002)

[LW96] Li, C., Wang, X.S.: A data model for supporting on-line analytical processing. In: CIKM, pp. 81–88 (1996)

[MZ09] Malinowski, E., Zimnyi, E.: Advanced Data Warehouse Design: From Conventional to Spatial and Temporal Applications, Data-Centric Systems and Applications, Incorporated 2nd corrected printing edn. Springer, Heidelberg (2009)

[PRP02] Pedersen, D., Riis, K., Pedersen, T.B.: A powerful and SQL-compatible data model and query language for OLAP. In: Australasian Database Conference (2002)

[SC09] Smith, B.C., Clay, C.R.: Microsoft SQL Server 2008 MDX Step by Step. Microsoft Press (2009)

[SHWC05] Spofford, G., Harinath, S., Webb, C., Civardi, F.: MDX Solutions. John Wiley & Sons, Inc., New York (2005)

An Approach to Defining Scope in Software Product Lines for the Telecommunication Domain

Radovan Cvetković[1] and Siniša Nešković[2]

[1] Telekom Srbija a.d., Technical Affairs Division,
Bulevar umetnosti 16a, 11000 Belgrade, Serbia
radovan.cvetkovic@telekom.rs

[2] University of Belgrade, Faculty of Organizational Sciences,
"Branislav Lazarevic" Laboratory for Information Systems,
Jove Ilica 154, 11000 Belgrade, Serbia
sinisa.neskovic@fon.bg.ac.rs

Abstract. The Next Generation Operations Systems and Software (NGOSS) is a solution framework for the development of Operations Support System/Business Support Systems (OSS/BSS) in telecom companies. This paper presents an approach to OSS/BSS building which is based on a specific combination of Software Product Lines Engineering (SPLE) and NGOSS. The focus of this paper is the first phase in SPLE which deals with the identification and scoping of software product families required to build an OSS/BSS. We present a generic architecture of required product families as well as a methodological procedure for their identification and scoping. Both are based on Enhanced Telecom Operation Map (eTOM), a process framework defined within NGOSS.

Keywords: Software Product Lines, Software Family, NGOSS, eTOM, OSS/BSS.

1 Introduction

The Operations Support System/Business Support Systems (OSS/BSS) represents a very complex information system which integrates business and technical subsystems of a telecommunication (telecom) company into a functionally coherent whole. TeleManagement Forum, an international telecom industry association, has developed a solution framework for the efficient building of OSS/BSS, which is called Next Generation Operations Systems and Software (NGOSS) [1]. Essentially, NGOSS represents a reference enterprise architecture framework for the telecom domain consisting of a set of reference models, methods and guidelines for OSS/BSS development. The Enhanced Telecom Operation Map (eTOM) is a business process reference model within NGOSS which identifies and categorizes all business activities that a telecom service provider will use. The eTOM framework supports two different perspectives of process groupings: 1) Horizontal process groupings represent a view of functionally related processes, which represent core business functions within the telecom business domain, and which are used as basic units in the automation of end-to-end processes; 2) Vertical process groupings represent a view of end-to-end processes within the business which effectively support customer needs in a total. These vertical end-to-end process groupings are essentially crosscutting overlays onto the hierarchical top level horizontal groupings.

Due to NGOSS complexity, a high level abstraction of its reference models and informal development methods and guidelines, a fruitful utilization of NGOSS is very hard to achieve in practice. This paper presents an approach to OSS/BSS development based on a specific combination of Software Product-Line Engineering (SPLE) principles and NGOSS. SPLE is a method which creates software product lines as development platforms for (largely automated) production of a family of software products [2], [3]. The main idea is to employ SPLE and exploit NGOSS for the development of a range of software product lines which are collectively capable of producing an OSS/BSS tailored to the specific needs of a particular telecom company, similarly to the idea expressed in [4]. The focus of this paper is the first phase in SPLE which deals with the identification and scoping of software product families.

The rest of the paper is organized as follows. In Section 2 we introduce a generic architecture of software product families required to build an OSS/BSS. A methodological procedure for their identification and scoping is given in Section 3. The paper concludes with the paper's main contributions and a discussion related to our future work.

2 Generic Architecture of Software Product Families

A proposed generic architecture of product families in the telecom domain is given in Fig. 1 as a metamodel in the form of an UML class diagram. Derived from the eTOM framework, the metamodel identifies the types of product families, represented as UML classes in the diagram, as well as their mutual relationships, represented as UML associations.

OSS/BSS class represents a type of product family whose instances are families used to produce individual OSS/BSS systems tailored to specific needs of a particular telecom company. Due to their complexity, OSS/BSS family members are not built as monolithic software applications, but as complex software systems composed of members from other product families supporting particular telecom domain aspects.

These families are structured following eTOM into two distinct types: *Business Domain* for family types which support core telecom business functions (eTOM horizontal processes), and *End-to-End process* for family types supporting eTOM vertical processes. Information about which particular families of *Business Domain* and *End-to-End Process* families constitute a particular *OSS/BSS* family member is captured by *Has Domains* and *Has Processes* aggregations. Similarly, which *Business Domain* families are used by a particular *End-to-End Process* family is captured by the *Uses* aggregation. It is important

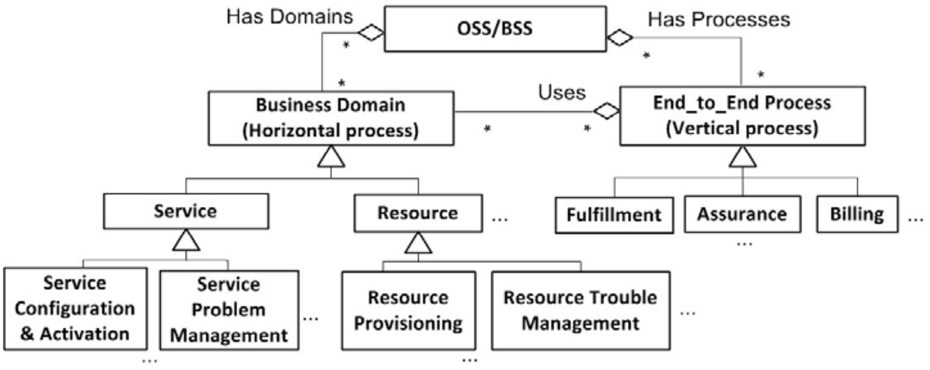

Has Domains
OSS/BSS
Has Processes
Uses
Business Domain (Horizontal process)
End_to_End Process (Vertical process)
Service
Resource
Fulfillment
Assurance
Billing
Service Configuration & Activation
Service Problem Management
Resource Provisioning
Resource Trouble Management

Fig. 1. Metamodel of Architecture of Software Product Families for the Telecom Domain

to observe that both *Business Domain* and *End-to-End Process* family types can be further specialized (subtyped) into family types supporting particular subdomains (illustrated in Fig. 1). Subtyping in this context means that each subtyped family type supports an additional functionality to its parent family type.

Since each family requires a separate product line for the production of its members, the generic architecture must be concretized (with classes and their instances) before the development of production lines is possible.

3 Identification and Scoping of Software Product Families

The concretization of the generic architecture is done through the identification and scoping of products families, defined as a separate process distinguished from the traditional Domain Engineering and Application Engineering processes [2].

Identification and scoping of software product families, given in Fig. 2 as an UML Activity Diagram, consists of the following activities:

– **eTOM to Feature Models Transformation** produces a set of feature models which are more convenient for further analysis
– **Software Product Families Type Identification** uses *eTOM feature models* to produce *Software Product Family Types Specification.* (i.e. it specifies classes in the metamodel of the generic architecture)
– **Software Product Families Identification** activity results with *Software product family specification* (i.e. it identifies instances of classes in the metamodel of the generic architecture)
– **Software Product Families Scope Definition** activity defines the high level scope for product families (i.e. defines commonalities of the identified product families)
– **Software Product Families Variability Definition** activity identifies the differences between members of the family and defines this variability

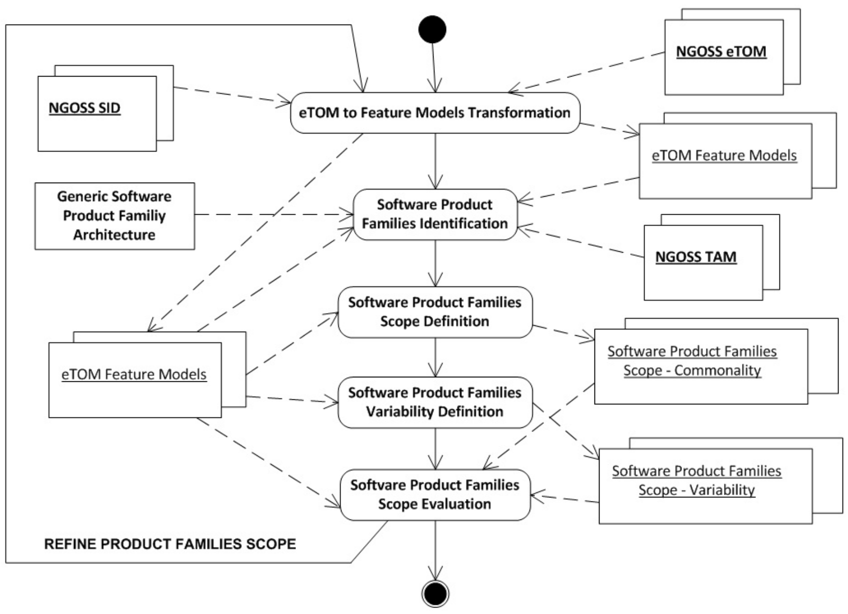

NGOSS eTOM
NGOSS SID
eTOM to Feature Models Transformation
eTOM Feature Models
Generic Software Product Familiy Architecture
Software Product Families Identification
NGOSS TAM
Software Product Families Scope Definition
Software Product Families Scope - Commonality
eTOM Feature Models
Software Product Families Variability Definition
Software Product Families Scope - Variability
Softvare Product Families Scope Evaluation
REFINE PRODUCT FAMILIES SCOPE

Fig. 2. Identification and Scoping Process

4 Conclusion

The presented approach augments traditional SPLE with: 1) the generic architecture of software product families for the telecom domain; 2) the method for the identification and scoping of product families.

The future work is related to the design and implementation of product lines for building these defined product families. The main challenge here is related to the realization of such an extremely complex software architecture consisting of a large number of mutually related product lines.

References

1. Reilly, J.P., Creaner, M.J.: NGOSS distilled: The Essential Guide to Next Generation Telecoms Management. TM Forum, UK (2005)
2. Pohl, K., Böckle, G., van der Linden, F.: Software Product Line Engineering Foundations, Principles and Techniques. Springer, Heidelberg (2005)
3. van Ommering, R.: Software Reuse in Product Populations. IEEE Trans. Software Eng. 31(7) (2005)
4. Bosch, J.: Using Software Product Families: Towards Compositionality. In: Dwyer, M.B., Lopes, A. (eds.) FASE 2007. LNCS, vol. 4422, pp. 1–10. Springer, Heidelberg (2007)

Stones Falling in Water: When and How to Restructure a View–Based Relational Database*

Eladio Domínguez[1], Jorge Lloret[1], Ángel L. Rubio[2], and María A. Zapata[1]

[1] Dpto. de Informática e Ingeniería de Sistemasm,
Facultad de Ciencias. Edificio de Matemáticas,
Universidad de Zaragoza. 50009 Zaragoza, Spain
{noesis,jlloret,mazapata}@unizar.es

[2] Dpto. de Matemáticas y Computación. Edificio Vives,
Universidad de La Rioja, 26004 Logroño, Spain
arubio@unirioja.es

Abstract. Nowadays, one of the most important problems of software engineering continues to be the maintenance of both databases and applications. It is clear that any method that can reduce the impact that database modifications produce on application programs is valuable for software engineering processes. We have proposed such a method, by means of a database evolution architecture (MeDEA) that makes use of database views. By using views, changes in the structure of the database schema can be delayed until absolutely necessary. However, some conditions oblige modifications to be made. In this paper we present an approach to detect *when* the restructuring process must be realized and *how* to carry out this restructuring process.

1 Introduction

The use of views can reduce the impact that database modifications produce on application programs, since changes in the database are delayed until absolutely necessary. However, it can happen that these changes ultimately become mandatory because of the problem of updatability of views. In the present paper, we describe an approach that, on the one hand detects *when* such changes are compulsory, and, on the other hand, indicates *how* to perform database maintenance using the tools and facilities provided by an extension of the MeDEA architecture [2,3]. The overall contribution means that database engineers are provided with an infrastructure that allows them to perform semi-automated evolution and maintenance tasks in such a way that the structure and extension of the database is altered only when absolutely necessary. This situation allows applications using these modified databases to continue functioning unchanged for a longer time period.

* This work has been partially supported by the Ministry of Science and Innovation, project TIN2009-13584, by the Ministry of Industry, Tourism and Commerce, project LISBioBank (TSI-020302-2008-8) and by the Government of Aragon, project LIS (PI108/08).

The rest of the paper is structured as follows. In the following section we present the MeDEA architecture and in Section 3 we explain when and how the database must be restructured. Finally, we outline further work.

2 The MeDEA Architecture

The contributions of this paper rely on the MeDEA architecture for database evolution we proposed in [2,3]. MeDEA is a metamodel–based database evolution architecture which has four components: *conceptual component*, *translation component*, *logical component* and *extensional component* (see Figure 1).

The way of working of MeDEA is as follows: given an EER schema in the conceptual component, the translation algorithm is applied to it and creates 1) a set of elementary translations in the translation component, 2) the relational database schema and 3) the extensional database schema (see the top part of Figure 1). When the data structure must be changed, the data designer issues the appropriate evolution transformations to the conceptual component. These changes are propagated to the rest of the components by applying a lazy and logical mechanism. The key aspect of this algorithm is that the old logical and extensional schemas remain unchanged, and the target schemas are not completely created but simulated. In general, when the conceptual evolution implies the creation of new elements (tables, attributes,...) they are in fact created, but the modification or elimination changes are simulated creating views. This fact is depicted in the central part of Figure 1. It can be seen in this figure that, after a forward evolution propagation process, the relational and extensional schemas include the old schemas, together with the 'Views' piece representing the views that simulate the modification and elimination changes and the 'New' piece representing the added schema elements.

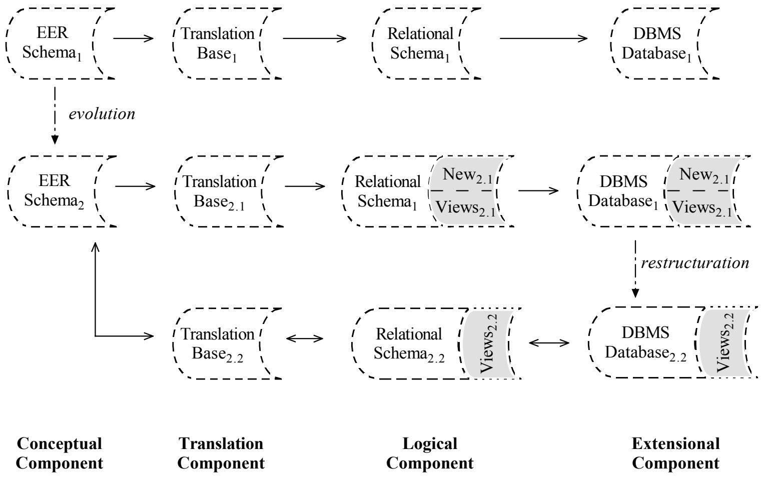


Fig. 1. View–based Evolution Processes within an EER–DBMS setting

Then, DML operations can be performed on the resultant extensional schema. In this case, the well-known problem of view updatability arises, that is, DML operations defined on a view cannot always be translated to the base tables [1,5], so that the schemas must be restructured. In order to solve this situation, we have identified *when* a database must be restructured in the presence of some particular DML operations and, on the other hand *how* this restructuring process can be carried out.

3 When and How to Restructure

Our first contribution is an answer to the following question: Under which conditions is it possible to translate operations on views into operations on base tables? This is the updatability problem whose question is to determine which operations on views can be translated into operations on base tables. For dealing with this question, we have decided to make the definition of updatability independent of the particular DBMS chosen, so we have defined updatability at the logical level. In this context, we have followed the solution of [5]. The paper [5] offers 'a theory within the framework of the ER approach that characterizes the conditions under which there exist mappings from view updates into updates on the conceptual schema'. It includes a set of definitions and theorems which determine, for entity type and relationship type views, whether they are insertable, deletable or updatable.

Our solution has been to adapt the view updatability algorithm of [5] to our particular context in which views are defined on the logical and on the extensional schema, unlike [5], where views are defined on the E/R schema. This adaptation can be handled because, in our evolution architecture, the conceptual and logical components are interconnected. So, we have defined a rule for managing external DML operations that determines *when* the restructuring is compulsory. The input of the rule is an external DML operation, defined on a view, that conforms with the conceptual schema. Then, the rule applies the adapted updatability algorithm to decide whether the operation is accepted. If accepted, the operation is executed provided that the integrity constraints are satisfied. If not, the extensional database must be restructured.

The second contribution is an algorithm that determines *how* the logical schema is modified and *how* the SQL code is generated in order to restructure the extensional database. We have chosen a backward-forward propagation strategy for this restructure algorithm because we can reuse the main elements of our architecture and, particularly, the translation component.

The input of the restructure algorithm is the view on which the DML operation is defined and then two tasks are interspersed. First, the conceptual elements related with the view are retrieved (backward propagation). Second, for each one of these conceptual elements its translation to the logical and extensional levels is modified (forward propagation). This translation is performed executing the procedure `considerApplyTranslRule` several times, which translates a conceptual element performing one of the three following possibilities:

1. A new translation rule is applied.
2. The same translation rule as before is reapplied but its effects will be different from the effects when it was previously applied.
3. Nothing is changed (the same translation rule would be applied but it would produce the same effect).

This restructuring process is depicted in the bottom part of Figure 1. The backward-forward process is represented by means of bidirectional arrows and the modifications performed in the schemas are represented changing the size of the pieces involved , in particular the 'View' piece is represented with a different size and shape.

A noteworthy characteristic of our algorithm is that the restructuring technique we propose only makes the compulsory changes so that the desired operation can be executed on the new extensional schema. This idea can be metaphorically compared with a stone falling in water. When a stone falls in water, the effect is propagated towards the shore in the form of several concentric waves. Similarly, the retranslation of a conceptual element is the stone which drops onto the schemas and this change generates concentric waves that carry the minimal changes towards other pieces of the schemas. For additional details, see [4].

4 Further Work

From here, several lines of work are opened up. Since we only use views at the logical level, we could analyze the impact that the use of views at the conceptual level would have on our overall proposal. Another distinct line, but related with the above, would be to introduce a query language at the conceptual level so that updatability problems could be addressed directly at this level.

References

1. Dayal, U., Bernstein, P.A.: On the Correct Translation of Update Operations on Relational Views. ACM Transactions on Database Systems 7(3), 381–416 (1982)
2. Domínguez, E., Lloret, J., Rubio, A.L., Zapata, M.A.: MeDEA: A database evolution architecture with traceability. Data & Knowledge Engin. 65(3), 419–441 (2008)
3. Domínguez, E., Lloret, J., Rubio, A.L., Zapata, M.A.: Model–Driven, View–Based Evolution of Relational Databases. In: Bhowmick, S.S., Küng, J., Wagner, R. (eds.) DEXA 2008. LNCS, vol. 5181, pp. 822–836. Springer, Heidelberg (2008)
4. Domínguez, E., Lloret, J., Rubio, A.L., Zapata, M.A.: Stones falling in water: When and how to restructure a view–based relational database (2010) (extended version), http://www.unizar.es/ccia/articulos.htm
5. Ling, T.W., Lee, M.L.: View Update in Entity-Relationship Approach. Data & Knowledge Engin. 19(2), 135–169 (1996)

Graph Object Oriented Database for Semantic Image Retrieval

Eugen Ganea and Marius Brezovan

University of Craiova, Craiova,
Bd. Decebal 107, Romania
{ganea_eugen,brezovan_marius}@software.ucv.ro

Abstract. This paper presents a new method for image retrieval using a graph object oriented database for processing the information extracted from the image through the segmentation process and through the semantic interpretation of this information. The object oriented database schema is structured as a classes hierarchy based on graph data structure. A graph structure is used in all phases of the image processing: image segmentation, image annotation, image indexing and image retrieval. The experiments showed that the retrieval can be conducted with good results and the method has a good time complexity.

Keywords: graph oriented object, object oriented database, image processing, image retrieval.

1 Introduction

Image retrieval systems have been developed using a variety of technologies in various disciplines of computer science. In this paper, we use the concepts of object-oriented programming for object recognition applications. In the Object-Oriented Databases ($OODB$), the relations is done by reference to an object identifier (OID) which is the key of association of the records. In identifying an unknown object, object recognition system queries the database and checking the similarities based on characteristics, between unknown object and each of objects from the database. We use a graph object-oriented model for $OODB$ representation based on a graph type data structure, where the operations on database objects are translated into the transformation of the graph. In addition the topological relations between the simple objects present in the image are represented by a graph edges, while the objects are the graph nodes. The construction of the graph model for image representation is based on a new utilization of pixels from the image that are integrated into a network type graph. We used a hexagonal network structure on the image pixels for representation of the graph $G = (V, E)$ and we considered two edges of graph joining the pseudo-gravity centers of the hexagons belongs to hexagonal network. For representing the output of the image segmentation process we used the Attributed Relational Graph (ARG) [4]. The result of segmentation algorithm is stored as a graph where the nodes represent the regions and the edges represent the neighborhood relations: G = (V_r, E_r), where V_r is the set of vertices corresponding regions detected and E_r is the set of edges that describes the neighborhood relations. The spatial relations between regions are divided into 3 categories: distance relations, direction relations and topological relations. For determining these types of relations we choose for each region the following relevant geometric features: the pseudo-center of gravity; the distance between two neighboring regions; the length of common boundary of two regions and the angle which is formed by two regions. The structure of paper is organized as follows: Section 2 describes the process of image annotation based on ontologies; Section 3 presents the graph object oriented database structure; Section 4 describes our experimental results and Section 5 concludes the paper.

1.1 Related Work

In this section we briefly consider some of the related work that is most relevant to our approach. A recent research [1] associate the labels with regions detected in the images training set, which poses a major challenge for learning strategy. They use a novel graph based semi-supervised learning approach to image annotation using multiple instances, which extends the conventional semi-supervised learning to multi-instance setting by introducing two level bag generator method. An explicit model named $GraphDB$ is presented in [2] and allows a simple modeling of graphs in an object oriented environment. The model permits an explicit representation of graphs by defining object classes whose instances can be viewed as nodes, edges and explicitly stored paths of a graph. In this paper we use, as the core of the proposed management system, the $HyperGraphDB$ [3], which is a database based on hypergraph structure and was development on $BerkleyDB$.

2 Image Annotation Based on Ontologies

In the image annotation process we use two types of ontologies: visual ontology which refer to an intermediate level which connecting lower level features to high level concept, and domain ontologies which refers to image content annotation. For annotate the simple objects we used learning algorithms based on decision trees - Decision Tree based Semantic Templates algorithm (DT-ST) [5]. DT-ST induction method for learning image semantics is different from classical algorithms that use semantic templates for continuous values of features of the regions. A ST feature is provided by the representative of a concept, the set of features extracted from the regions of the training images. Built the decision tree to assign high-level concepts, which are attached to leaf nodes of the tree, to lower level features, thus each useful concept of ontology will meet at least one leaf node. In the system developed, each image is divided into a number of areas which can attach semantic meaning, and each extracted region have as member, an instance of class $CFeatureVector$. For each concept we consider a vector with 7 components [$H\ S\ V\ perimeter\ compactness\ eccentricity\ area$], whose normalized values are used to construct the decision tree; the obtained decision tree is translated in a system of rules. The graph grammars were first used for image representation and then its were used for the syntactic representation and analysis of images are defined the spatial relationships between regions of an image. Grammar induction system for a type graph, $SubdueGL$ algorithm was developed based on $Subdue$ [6] and uses a breakthrough approach to sub-graphs. Using a growth process graph,$SubdueGL$ generate candidate sub-structures can be used to compress the data set of the graph.

3 Graph Object Oriented Database

The semantic information correspond to concepts of domain ontology on the one hand and to elements of visual ontology on the other hand. The visual concepts determined automatically in the phase of post-processing of segmentation results are stored implicitly in the ARG structure representing relations between regions of an image. Each semantic object will have an attribute that points to the object region interpreted (OID). The OID of semantic object is the same with the identifier of the corresponding synset from $WordNet$ [7]. In this way, we manage the uniqueness of OID attribute values and we provide the link for an annotation with different concepts, but there is a relationship between synonyms. Indexing problem is approached using graph theory, the relationship is represented by indexing the index assign classes and forming a directed graph. Based on the database scheme, was developed a new approach to the problem of indexing by exploiting graph structure type; this type uses an index nested/inherited. Using facet $index - propagation$ attached to each attribute indexes are grouped according to the characteristics of each object processed region. For the index management of the $OODB$ we build a system of indexes based on the geometric and the semantic attributes of the shapes. As a result of our tests we consider two groups of indexes; the first group (geometric group) is used only for the training images in the off-line phase of system utilization - learning phase, and the second group (semantic group) is used for all other images in the online phase of system utilization - symbolic query phase. First the group of indexes belongs to the attributes extracted after the image segmentation: the perimeter, the gravity center, compactness of shape, eccentricity of shape, the list of gravity centers of hexagons from the contour and the syntactic characteristics of the boundary shape. This approach drives at a good optimization of the retrieval process for linking an image region to a synset. In this stage the $OODB$ contains only the information corresponding to the ontology so the space taken by the system of indexes has not influence concerning the storage performance. At the end of this phase the first group of indexes is deleted. The query expressions written in symbolic language must be analyzed and converted to an equivalent native query format. In this process the relationships between the concepts of the ontology on one part and between concepts and classes on the other part are used. For all the words present in the query expression we search the correspondence with the synsets from the $WordNet$ taxonomy and mark these synsets. In the second step for each returned synset from the list after the first stage we determine the corresponding class and we make an instance for the class through the call of the constructor which receives the name of the synset. After the execution, a native query is obtained in this mode, and we have a list of objects corresponding to the images with semantic content according to the query expression.

4 Experiments

A prototype system was designed and implemented in $Java$, and $HyperGraphDB$. We tested our system on $Princeton\ Event$ dataset [8]. The retrieval process implies the experiments for the retrieval process based on symbolic language queries. In this case there were taken into consideration the pseudo-natural queries based on the concepts from ontology. We use the *Princeton Event* dataset which contains 8 sport activities. For each category we consider 25 representatives images for the learning phase. In the *OODB* "sports.hgdb" are stored initially the information extracted by the segmentation from the training images. Using the indexes as the perimeter, the pseudo-gravity center, compactness of shape, eccentricity of shape and the list of gravity centers of hexagons from the contour we allocate and store in the sports.hgdb all images corresponding to the dataset. After the learning and storing phase the OODB is ready to be interrogated. The pseudo-natural query considered is: *red ball one hoop* Using the data from the ontology and the *Wordnet* information, this query is translated in equivalent object oriented native query: *CImage imageQuery* = *new CImage* ("*croquet with red ball and one hoop*")

5 Conclusions

In this paper, we propose a method for image processing based on graph structure with the aim of good retrievals. The process have three phases: (I) an image segmentation based on graph structure; (II) an adaptive visual feature object-oriented representation of image contents; and (III) a management of the ontologies uses for annotation. Using these tree stages and an object-oriented wrapper for *HyperGraphDB*, the system allow two queries based on symbolic language. The future work implies the description and the using of the graph grammar with the goal of searching and retrieving complex images based on the complex query formulated in a symbolic language.

Acknowledgment. This research was supported by CNCSIS - UEFISCSU under a grant PNII-IDEI 535/2008.

References

1. Hu, X., Qian, X.: A Novel Graph-based Image Annotation with Two Level Bag Generators. In: International Conference on Computational Intelligence and Security, vol. 2, pp. 71–75 (2009)
2. Guting, R.H.: GraphDB: Modeling and Querying Graphs in Databases. In: Proceedings of 20th Int. Conf. on Very Large Data Bases, pp. 297–308 (1994)
3. HyperGraphDb, `http://www.kobrix.com/hgdb.jsp` (consulted 01/02/2010)
4. Hong, P., Huang, T.S.: Spatial pattern discovery by learning a probabilistic parametric relational graphs. Discrete Applied Mathematics 139, 113–135 (2004)
5. Liu, Y., Zhang, D., Lu, G., Tan, A.: Integrating Semantic Templates with Decision Tree for Image Semantic Learning. In: Cham, T.-J., Cai, J., Dorai, C., Rajan, D., Chua, T.-S., Chia, L.-T. (eds.) MMM 2007. LNCS, vol. 4352, pp. 185–195. Springer, Heidelberg (2006)
6. Holder, L.B.: Empirical Substructure Discovery. In: Proceedings of the Sixth International Workshop on Machine Learning, pp. 133–136 (1989)
7. Miller, G.A.: Nouns in WordNet: a Lexical Inheritance System. International Journal of Lexicography 4, 245–264 (1990)
8. Li, L.-J., Fei-Fei, L.: What, where and who? Classifying event by scene and object recognition. In: IEEE International Conference in Computer Vision, ICCV (2007)

Natural Language Querying over Databases Using Cascaded CRFs

Kishore Varma Indukuri, Srikumar Krishnamoorthy, and P. Radha Krishna

SETLabs, Infosys Technologies Limited, India
{Kishore_Varma,Srikumar_K,RadhaKrishna_P}@infosys.com

Abstract. Retrieving information from relational databases using a natural language query is a challenging task. Usually, the natural language query is transformed into its approximate SQL or formal languages. However, this requires knowledge about database structures, semantic relationships, natural language constructs and also handling ambiguities due to overlapping column names and column values. We present a machine learning based natural language search system to query databases without any knowledge of Structure Query Language (SQL) for underlying database. The proposed system - Cascaded Conditional Random Field is an extension to Conditional Random Fields, an undirected graph model. Unlike traditional Conditional Random Field models, we offer efficient labelling schemes to realize enhanced quality of search results. The system uses text indexing techniques as well as database constraint relationships to identify hidden semantic relationships present in the data. The presented system is implemented and evaluated on two real-life datasets.

1 Introduction

The approaches taken to address such natural laguage queries can be broadly classified as template based [2][6], rule-based [3][4], and machine learning [5] based models. These systems aim to capture the patterns in the natural language effectively to provide an efficient semantic search system. This paper provides a machine learning algorithm which is a Cascaded CRFs based approach that extends a traditional CRF model to label search terms efficiently using semantic relationships present in underlying database structures. The presented system also employs text indexing techniques to identify latent semantic relationships and improve the quality of search results.

Inference and Scoring Using CRF's

Names assigned to tables in a database may intentionally or unintentionally coincide with those given to columns in the same table or other tables. Therefore, many tools that work on lexicon look-up (mainly used in rule-based approaches) fail to distinguish between the names (column or table names) with their table contents (values). The proposed approach attempts to solve this problem with

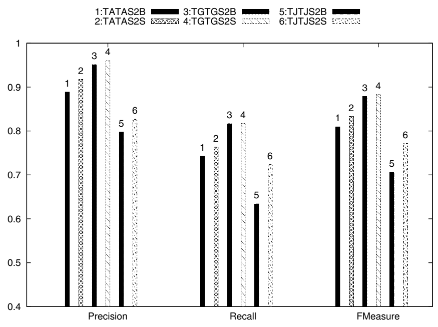


Fig. 1. Comparing results of entire dataset with individual Geo and Job Query datasets

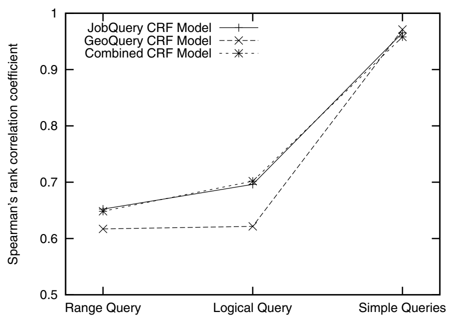


Fig. 2. Spearman's rank correlation coefficient obtained for top 10 search results

Table 1. Results for combined dataset with individual Geo and Job Query datasets

CRF Name	P	R	FM	Train (sec)	Test (sec)
TATAS2B	.889	.743	.810	214	7
TATAS2S	.917	.763	.833	366	7
TGTGS2B	.951	.817	.879	94	3
TGTGS2S	.960	.817	.883	101	3
TJTJS2B	.798	.634	.707	113	4
TJTJS2S	.827	.723	.771	220	5

Table 2. Results for Cascaded CRF, Dynamic CRF with linear-chain clique template for *CrfS1*

Type	CRF Name	P	R	FM	Train (sec)	Test (sec)
DCRF	TATAS2BM	.885	.688	.775	1681	42
DCRF	TATAS2BMS	.887	.684	.773	2047	51
CCRF	TATAS2B	.897	.632	.741	228	7
CCRF	TATAS2S	.901	.733	.808	301	7

the use of machine learning algorithms. We use two CRFs applied sequentially to the sequence of words in the user query. The first CRF (*CrfS1*) is trained to label each of the instance with state labels to identify column names (*ColName*) and column values (*ColVal*) from the user's natural language query. The state which is not *ColName* and *ColVal* is labelled with tag *Other*. The second CRF (*CrfS2*) is trained to take as input the instance along with its label sequence assigned using *CrfS1* and predict the two nodes which can be connected using a skip-edge and label them with *Start* and *End* tags. The skip-edge does not impose any constraints on the order of the tags which it is connecting. i.e. *Start* node can be either column name node or column value node. Those states which are neither *Start*, nor *End* are tagged as *Oth*.

The scores that can be used for sorting the results based on relevance with the users natural language query ($\mathbf{x}$) is a product of accuracies obtained in linear-chain (*CrfS1*), skip-chain (*CrfS2*) and the similarity of words in $\mathbf{x}$ labelled as column names and column values with the actual database table column names and values. The final term in the product penalizes the poor naming convention followed while designing the database which makes differentiating column names, table names and their values nearly impossible. If $P_l(\mathbf{y}|\mathbf{x})$, $P_s(\mathbf{y}|\mathbf{x})$ represent

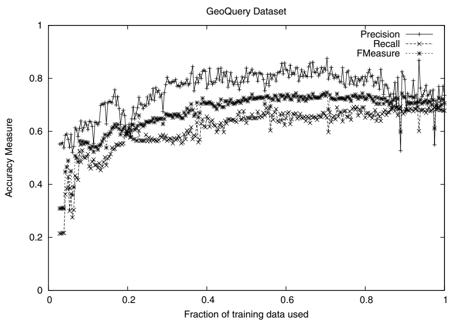


Fig. 3. Effect of data size on Accuracy with GeoQuery Dataset

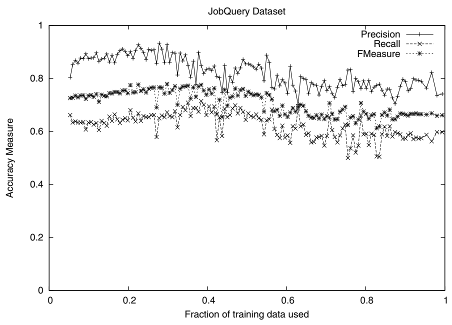


Fig. 4. Effect of data size on Accuracy with JobQuery Dataset

linear chain and skip chain conditional probabilities obtained for input state sequence $\mathbf{x}$. The combined accuracy $P(\mathbf{y}|\mathbf{x})$ of the Cascaded CRF model can be written as $P(\mathbf{y}|\mathbf{x}) = P_l(\mathbf{y}|\mathbf{x}) \times P_s(\mathbf{y}|\mathbf{x})$.

If $IndexerScore_i$ denotes the search result score obtained from an indexer for a particular search hit, the altered score which includes credit for a match in the functional relationship in that search hit is given as $Score_i = W_{S1} + W_{S2} + IndexerScore_i$. During offline indexing, for evaluation of the search application developed over a database with the help of training data represented as set $\{\mathbf{x}\}$, for each of the instance, x_i in this training set, if $\hat{x_i}$ represents bag of words extracted as *ColName* and *ColVal* which includes terms labelled *Start* and *End*, and if $J(\hat{x}, I)$ represents the similarity measure of the words in $\hat{x}$ with the meta information[1], I from the database, the final accuracy of the application generated using the CRF model built with training data over a particular database is:

$$accuracy = \prod_{x_i \in \{\mathbf{x}\}} P(\mathbf{y}|x_i) \times \frac{J(\hat{x_i}, I)}{\sum_{t \in I} J(t, I)} \tag{1}$$

From Equation 1, it can be inferred that as the similarity among various column names and table names increases, the accuracy of overall scoring function for that particular database decreases. The database with least value in the denominator for the Equation 1 will yield maximum accurate (confident) results.

2 Experimental Results

Two commonly used real-life datasets JobQuery and GeoQuery datasets[2] [1] are used for experimentation. The naming convention used according to alphabet position is : first alphabet T for Training Data, second alphabet J/G/A for JobQuery or GeoQuery or Both(All) respectively, third alphabet T for Testing

[1] Meta-Information include names given to the table, columns.

[2] Available from Machine Learning website of University of Texas.

Data, fourth alphabet J/G/A for JobQuery or GeoQuery or Both(All) respectively, 5 and 6 alphabets S2 for Stage 2 (Labelling with *Start*, *End* and *Oth* tags) and 10 , 11 and 12 alphabets take values B/S/M which mean Bigram Template/ Skip-Chain Template/ Mapping Template respectively. Several experiments were conducted by varying the parameters like training dataset, testing dataset, labeling scheme / templates used etc. Each experiment is uniquely named using the parameter naming convention details as outlined above.

The efficacy of the proposed Cascaded CRF approach is assessed using three set of experiments: First set of experiments (Figure 1) compare skip chain CRF with linear chain CRF. The CRF versions use ideally tagged training data for *CrfS2* in place of tagged data from *CrfS1*. The experiments are conducted for each of the GeoQuery, JobQuery and combined datasets. For obtaining ideally tagged data, we identify *ColName* and *ColVal* entities in the training data with the help of formal language equivalent for the natural language query available for each of the instance in the dataset. Second set of experiments (Table 2) compare Cascaded CRF with DCRF. Experiments on Cascaded CRF and DCRF are repeated with linear-chain and skip-chain clique templates for *CrfS2* and second layer of DCRF respectively. In order to show the effect on final ranking of search results, we have compared the order of results generated by the system with the order of results tagged manually for a defined set of natural language queries. "Spearman's rank correlation coefficient" for top 20 search query results are shown in Figure 2. Third set of experiments (Figure 3 and Figure 4) provide the sensitivity analysis of quality of results for varying training data instances.

References

1. Jayapandian, M., Jagadish, H.V.: Automated creation of a forms-based database query interface. PVLDB 1(1), 695–709 (2008)
2. Mador-haim, S., Winter, Y., Braun, A.: Controlled language for geographical information system queries. In: Proceedings of Fifth International Workshop on Inference in Computational Semantics (2006)
3. Popescu, A.-M., Armanasu, A., Etzioni, O., Ko, D., Yates, A.: Modern natural language interfaces to databases: composing statistical parsing with semantic tractability. In: COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics, Morristown, NJ, USA, vol. 141. Association for Computational Linguistics (2004)
4. Popescu, A.-M., Etzioni, O., Kautz, H.: Towards a theory of natural language interfaces to databases. In: IUI 2003: Proceedings of the 8th International Conference on Intelligent User Interfaces, pp. 149–157. ACM, New York (2003)
5. Sutton, C., McCallum, A., Rohanimanesh, K.: Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. J. Mach. Learn. Res. 8, 693–723 (2007)
6. Thompson, C.W., Ross, K.M., Tennant, H.R., Saenz, R.M.: Building usable menu-based natural language interfaces to databases. In: Schkolnick, M., Thanos, C. (eds.) VLDB, pp. 43–55. Morgan Kaufmann, San Francisco (1983)

A Data Mining Design Framework - A Preview

Kai Jannaschk and Tsvetelin Polomski

Christian-Albrechts-University Kiel, Information Systems
Kiel, Germany
{kaja,tpo}@informatik.uni-kiel.de

Abstract. Data storages contain a lot of hidden information unknown to their owners. There are many different types of data mining processes, which aim to provide the means to expose this hidden information. But existing data mining processes mainly illustrate the management process and not really the discovery process. The user still has to decide, which methods and algorithms to apply. Furthermore, correct interpretation of the result can be challenging. In this paper we describe a framework for a systematic knowledge discovery process, which is split into three stages. We define the requirements, methods, and possible outcomes for each stage.

Keywords: knowledge discovery, data mining, process design.

1 Introduction

Data mining is the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns from data.

Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth [2]

This definition of data mining describes only a part of a knowledge discovery process. The community uses the terminology Knowledge Discovery and Data mining (KDD) when they want to refer to the process as a whole. There is a concrete data set, and we are searching for a data independent pattern. Patterns express the knowledge behind the data.

2 A Data Mining Design Framework

A well known knowledge discovery process is the CRISP-DM process ([1]). It is split into six phases. Two different groups of experts are collaborating. One group are the domain experts, which have knowledge in the domain, and can explain the discovered models. The other group are technical experts, who have the knowledge in the usage of data mining techniques. The CRISP DM process mainly illustrates the management process, and not really the discovery process. The user knows the goal of the process. He doesn't know, which requirements the tasks have, and what he can do with the output.

There are many different data mining algorithms, each with different requirements for data quality, different goals, and of course different parameters. A knowledge discovering user has to know all the algorithms, and to choose one for his/her concrete problem.

We see not only in CRISP DM ([1]), that there are many different phases in such a process. The process is iterative and the user makes decisions, choosing the next steps in the process. In figure 1 we illustrate a framework, that contains three levels of our understanding of a knowledge discovery process. Following we describe each stage.

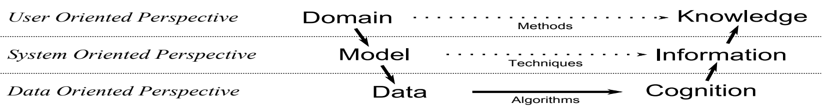

User Oriented Perspective
Domain
Methods
Knowledge
System Oriented Perspective
Model
Techniques
Information
Data Oriented Perspective
Data
Algorithms
Cognition

Fig. 1. Data Mining Design Framework

2.1 User Oriented Perspective

The basis of this part of the process is the user knowledge, or domain knowledge. We introduced a triangle to the information science as a relation between *concept*, *topic*, and *content* ([3]). The concepts describe appearance and behavior of things and relations in the real world. They declare a field of facts. We describe content in a variety of ways. The topics describe the content, and the meaning of the topics is defined by the concepts. So the same topic can be used in different concepts with different meaning. Finally, the content provides the facts for the concepts, and is described by the topics. The content represents aspects of the real world. For [4] is it important, to realize the influence between the conceptual background and the real things.

A *supposition* is an assumption, which can be proven by empirical or theoretical methods. It's an observation in the content or a phenomenon in the concepts. The aim in our process is to validate such a supposition. Is the knowledge about the concepts for the content sufficient and valid? Or does the content fit the concepts assigned to the given topic?

A scientific theory represents a supposition, or a group of related suppositions, which has been confirmed through repeated experimental tests or methods. Accepted scientific theories become part of our understanding, and are the basis for exploring less well-understood areas of knowledge. New discoveries are first assumed to fit into the existing conceptual framework. Only when the new phenomenon cannot be accommodated, scientists seriously question the concepts and attempt to modify it.

Thus we characterize the *knowledge*, based on a scientific theory. We know, that the concept space of the human race is incomplete ([4]). The human influence on the characteristics of knowledge is very high. The worth of the knowledge is dependent of the necessity and the real prior knowledge of the human.

2.2 System Oriented Perspective

The second level represents the interface between the domain knowledge and the starting point of the technical process. Based on the domain knowledge we create a model. A *model* summarizes the concepts above. A model is a system of assumptions based on some experimental facts sometimes containing adjustable parameters. The topics are the *properties* of a model and the data. The set of "right" properties depends on the point of view, or rather on the topics under which the considered objects are being explored. The content is source of possible *data* for the model. We consider each formation of symbols as data as long as it represents information according to some convention. Thus, when speaking about data, one has to consider both the *symbols*, representing the data, and the *meta-data*, expressing the meaning of those symbols or their particular usage. The aim is to validate suppositions of model and data by different techniques. This validation will be done by different data, and will result in a superset of the starting model, which contains new *information.*

2.3 Data Oriented Perspective

On the third level of our knowledge discovery process we use known algorithms and methods for data analysis. At this point real data sets are being analyzed in order to draw conclusions about the content. A high-level, global description of a data set is given by a *schema* based on a set of attributes. The attributes are derived from properties. The schema declares the kinds of relations that might exist between the attributes, as well as additional meta information such as why, when and by whom the data set was constructed.

A KDD process at this stage aims at the verification of suppositions on a technical level. For a data set $\mathcal{D}$ and its schema $\mathcal{S}$, a possible supposition $\mathcal{H}$ might be e.g. that $\mathcal{D}$ contains artifacts not specified by $\mathcal{S}$. A tuple $L_i := (\mathcal{S}, \mathcal{D}, \mathcal{H})$ containing those three elements is the starting point of a process. Since each KDD process is working on a set of tuples of this kind, we define the KDD process elements as a set $L := \{(\mathcal{S}, \mathcal{D}, \mathcal{H})\}$. We assume, that the result of the process is a set L_o of tuples $(\mathcal{M}, q)$, containing a model $\mathcal{M}$ with a quality q. The agents $\{Ag\}$ participating in the process describe functions on L and thus, an agent is defined as $Ag : L \to L$, $l \mapsto l'$. We differentiate between four types of agents, each of them having its own specifics:

1. *data preparation agents* Ag_w: $(\mathcal{S}, \mathcal{D}, \mathcal{H}) \mapsto (\mathcal{S}, \mathcal{D}', \mathcal{H})$, preparing the data set for the following steps (e.g. outlier detection)
2. *data exploration agents* Ag_e: $(\mathcal{S}, \mathcal{D}, \mathcal{H}) \mapsto (\mathcal{S}, \mathcal{D}, \mathcal{H}')$, creating suppositions for validation (e.g. data visualization techniques)
3. *descriptive agents* Ag_d: $(\mathcal{S}, \mathcal{D}, \mathcal{H}) \mapsto (\mathcal{S}', \mathcal{D}, \mathcal{H})$, working on the schema describing the data set (e.g. cluster analysis methods).
4. *predictive agents* Ag_p: $(\mathcal{S}, \mathcal{D}, \mathcal{H}) \mapsto (\mathcal{M}, q)$, creating the models (e.g. a classification algorithm using the clustering result to construct a model predicting the class association for new objects).

With the terms discussed so far, a *KDD process* is a tuple $(L_i, \{s_j\}, L_o)$ where $\{s_j\}$ is a set of steps, each of them being performed by an agent. Additionally, a process has the following characteristics: 1. A process starts with one tuple of L; 2. each tuple of L can be input for one or more agents; 3. the predictive agents Ag_p are the last agents in a process branch; 4. data preparation agents Ag_w can be employed anytime; 5. exploration agents Ag_e can be used anytime before descriptive Ag_d and predictive Ag_p agents.
We also have an explanation model Φ, showing the transformation of given input into an outcome model. Thus, we define Φ as the function $\Phi : 2^L \times L_i \longrightarrow L_o$. This way the user knows what kind of agents were used for constructing the model.

Finally, the sketched framework can be regarded as a tuple $(L, L_i, L_o, \{Ag\}, \Phi, C)$ consisting of the already explained components, and additionally, of a controller C, recommending at each state a set of appropriate agents. The controller uses the possible models to reach a validation of the stated supposition.

3 Conclusion

Our framework provides three different levels of abstraction of knowledge and can be applied with many different algorithms and methods for analyzing data. The framework supports the user in interpreting the resulting knowledge.

We start out with the first level, the User Oriented Perspective, where the knowledge of a user is the focus of the process. Methods for knowledge creation at this stage are provided. On the second level, the System Oriented Perspective, we present an interface between the domain knowledge of a user, and the technical data analysis algorithms as a method for knowledge creation. In the third stage, the Data Oriented Perspective, we define a process assisted by agents, working on a set of elements. Each element contains a schema $\mathcal{S}$ describing a real data set $\mathcal{D}$, and a supposition $\mathcal{H}$, which the user wants to validate. The result of this process is a model $\mathcal{M}$ with a quality q. The path from input to result is described by an explanation model Φ.

References

1. Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., Wirth, R.: CRISP-DM 1.0 Step-by-step data mining guide. In: SPSS, NCR, DaimlerChrysler (2000)
2. Fayyad, U., Piatetsky-Shapiro, G., Smyth, P.: From Data Mining to Knowledge Discovery in Databases. AI Magazine 17, 37–54 (1996)
3. Kidawara, Y., Zettsu, K., Kiyoki, Y., Jannaschk, K., Thalheim, B., Linna, P., Jaakkola, H., Du, M.: Knowledge Modeling, Management and Utilization Towards Next Generation Web. In: Information Modelling and Knowledge Bases XXI. IOS Press, Amsterdam (2010)
4. Murphy, G.L.: The big book of concepts. MIT Press, Cambridge (2001)

On Support of Ordering in Multidimensional Data Structures*

Filip Křižka, Michal Krátký, and Radim Bača

Department of Computer Science
VŠB-Technical University of Ostrava, Czech Republic
{filip.krizka,michal.kratky,radim.baca}@vsb.cz

Abstract. Multidimensional data structures are applied in many areas, e.g. in data mining, indexing multimedia data and text documents, and so on. There are some applications where the range query result must be ordered. A typical case is the result with tuples sorted according to values in one dimension defined by the `ORDER BY` clause of an SQL statement. If we use a multidimensional data structure, the result set is sorted after the range query is processed. Since the sort operation must often be processed on tuples stored in the secondary storage, an external sorting algorithm must be utilized. Therefore, this operation is time consuming especially for a large result set. In this paper, we introduce a new data structure, a variant of the R-tree, supporting a storage of ordered tuples.

Keyword: name multidimensional data structures, ordered tuples of the range query result, R-tree, Ordered R-tree.

1 Introduction

Multidimensional data structures [8] have been widely applied in many data management areas. Indexing spatial data is their natural application but there are many applications in various domain areas. Multidimensional data structures may be applied as index data structures supporting the storage of tuples in relational or object-relational DBMS's [2]. If the B-tree or the B^+-tree are applied for the support, we must create one tree for each attribute to be indexed. Another alternative is to use the so-called compound-key including more attributes. In this case, some queries are processed by an often inefficient sequential scan. When we consider multidimensional data structures, one index is created for all attributes to be indexed. In this case, random accesses may influence the efficiency of query processing [5]. The efficiency of range query processing in the B-tree is also influenced by this issue.

There are some applications where ordering of tuples in the result set is required. This order may be defined by ordering of values of one attribute. A common example is an SQL statement with the `ORDER BY` clause. We often define a restriction of attribute values, which means the range query can be applied for

* Work is partially supported by Grant of GACR No. P202/10/0573.

the processing of this query; however, we often require an ordering defined by the `ORDER BY` clause. If we consider multidimensional data structures, there is no support for the ordering; the result tuples must be sorted after the query is processed. Since the number of sort operations and the number of tuples in the result may be high, we must utilize an external sorting algorithm [4,10]. If large result sets are sorted, this operation is time consuming.

When we consider multidimensional data structures without a support of ordering, a lazy evaluation is not possible; the query must be processed in one and result tuples must be inserted in a persistent data structure and sorted [6]. In the case of lazy query processing, a query is progressively processed and we can not store all tuples of the result in an additional data structure.

In this article, we introduce a multidimensional data structure with a support of ordering. A cost-based relational query optimizer can simply utilize this data structure [2]. Since the R-tree is the most popular data structure we present the support of ordering for the R-tree. Although it can be seen that there is a relation between multidimensional data structures and ordering, we only find data structures which enable the indexing of multidimensional data using an order. Such data structures include *UB-tree* [1] which utilizes Z-ordering for indexing of multidimensional data. When we apply B-tree or B$^+$-tree, a range query is processed by an often inefficient sequential scan.

2 Ordered R-Tree

In the R-tree [3], tuples are clustered in tree's pages when *MBRs* (*Minimal Bounding Rectangles*) are built. R-tree supports various types of queries, e.g. point and range queries. A range query is processed by a depth-first search algorithm. If the query rectangle intersects an MBR then the node related to the MBR is retrieved and searched. In general, there is no order; therefore, tuples of a result set are sorted in the same order in which these tuples were inserted in leaf nodes and new MBRs have been inserted in inner nodes. When we consider the previously depicted motivation, the `ORDER BY` clause, we want to retrieve tuples under the well-known Lexicographical order of tuples. This order is presented by the dashed line in Figure 1(a).

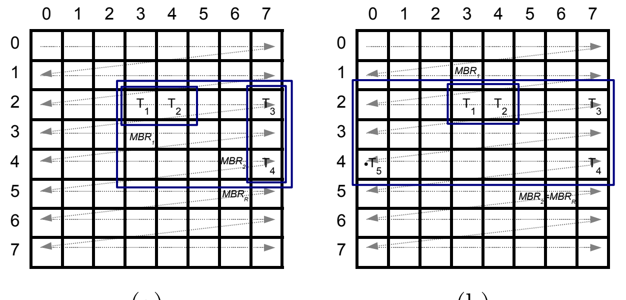


Fig. 1. Examples of the (a) ordered result set (b) unordered result set

Let us suppose the MBR_R in Figure 1(a) including tuples T_1–T_4. We use the order which prefers values of the first dimension before values of the second dimension. A range query containing MBR_1 and MBR_2 returns the following ordered result T_1,T_2,T_3,T_4.

After the tuple T_5 is inserted, MBR_2 must be enlarged (see Figure 1(b)). In this case, ${Q_L}^{MBR_1} = (2,3)$, ${Q_L}^{MBR_2} = (2,0)$. A range query containing both MBRs returns the following unordered result T_3,T_5,T_4,T_1,T_2, although tuples in both MBR's are ordered. It means, we can not utilize Q_L to the guarantee of the order.

It is clear that the Lexicographical order is equivalent to C-ordering. Another space filling curve [7], Z-curve, is utilized by the UB-tree [1]. There are two ways how to develop a data structure supporting an order. The first way is to create regions as intervals of the order used. The main problem of this way is that it is necessary to develop an algorithm checking whether the MBR and the regions are intersected. In [1,9] authors introduced this algorithm for the UB-tree and Z-ordering. The similar algorithm must be implemented for each order used. The second way is to use a known data structure and change the build algorithm to guarantee the ordered result set.

To support the order we add a special tuple, so-called *First Tuple* (FT), to each MBR of the R-tree. All nodes are then ordered according to their FT. First tuples are utilized during the tree building. We use a persistent array to manage first tuples.

3 Experimental Results

In our experiments[1], we used the real collection of meteorological data including 1,391,049 tuples of dimension 6 (see `http://portal.chmi.cz/`). All data structures have been implemented in C++. We built 2 data structures: R*-tree and Ordered R-tree (see Table 1(a)), the page size 2,048 B. Data structures have been built by a common tuple-by-tuple way. Since the R-tree and the Ordered R-tree create different regions, both build times are different for the same order of the inserted tuples. In our experiments, we used 30 range queries with various selectivities; the result set includes 0–548,725 tuples.

As usual, tests are processed with a cold cache (OS cache as well as cache buffer of indices). For all tests we measure query processing time and Disk Access Cost (DAC). Moreover, we measure the sorting time and DAC of the sort operation. Since the range query is processed by random disk accesses, DAC is equal to the number of disk accesses during the query processing [6]. We used Merge sort [4] for sorting of the result set. For the measurement of the query processing time, we repeat the tests 10$\times$ and calculate the average time.

DAC results are shown in Table 1(b). We see that the DAC for the Ordered R-tree is 125% of DAC for the R*-tree. Although DAC for the range query and sort operation are not comparable due to the fact that it is simply possible to reduce the number of disk accesses using a cache buffer in the case of the sort operation, we see the DAC of the sort operation is enormous especially for queries with a large result set.

[1] The experiments were executed on an Intel® Core 2 Duo 2.4 Ghz, 512 kB L2 cache; 3 GB RAM; Windows 7.

Table 1. (a) R-trees statistics and (b) DAC for tested range queries

	R*-tree	**Ordered R-tree**
Tree Height	4	4
Build Time [s]	233	159
# Inner Nodes	1,318	1,372
# Leaf Nodes	27,826	29,663
Index Size [kB]	58,290	62,072
FT Index Size [kB]	–	794

Query	**R*-tree**		**Ordered R-tree**
	Range Query	Sort	
$\mathbb{Q}_1$	2,279	17,408	1,545
$\mathbb{Q}_5$	201	205	221
$\mathbb{Q}_8$	7,600	31,795	5,372
⋮			
Avg.	6,297	56,627.2	8,478

Consequently, the average query processing time for R*-tree is 8.1 s (including 2.8 s for sorting), the average time for Ordered R*-tree is 5.2 s.

4 Conclusion

In this article, we introduced a new data structure, a variant of the R-tree, supporting a storage of ordered tuples. In this way, the range query result is ordered without an application of any time consuming sort operation. Our experiments show an improvement of this data structure in comparison with the R-tree. Orderer R-tree saves 35% of the query processing time compared to the R*-tree. The new data structure is particularly efficient in the case of large result sets or in the case of lazy query processing when tuples are not immediately retrieved. In our future work, we want to adopt this data structure for indexing XML data where the lazy query processing of ordered tuples can be applied.

References

1. Bayer, R.: The Universal B-Tree for multidimensional indexing: General Concepts. In: Masuda, T., Tsukamoto, M., Masunaga, Y. (eds.) WWCA 1997. LNCS, vol. 1274. Springer, Heidelberg (1997)
2. Garcia-Molina, H., Ullman, J., Widom, J.: Database Systems: The Complete Book. Prentice-Hall, Englewood Cliffs (2002)
3. Guttman, A.: R-Trees: A Dynamic Index Structure for Spatial Searching. In: Proceedings of ACM SIGMOD 1984, Boston, USA, pp. 47–57. ACM Press, New York (1984)
4. Knuth, D.: The Art of Computer Programming, 2nd edn. Sorting and Searching, vol. 3. Addison-Wesley, Reading (1998)
5. Lahdenmäki, T., Leach, M.: Relational Database Index Design and the Optimizers. John Wiley and Sons, New Jersey (2005)
6. Lightstone, S.S., Teorey, T.J., Nadeau, T.: Physical Database Design: the Database Professional's Guide. Morgan Kaufmann, San Francisco (2007)
7. Sagan, H.: Space Filling Curves. Springer, Heidelberg (1994)
8. Samet, H.: Foundations of Multidimensional and Metric Data Structures. Morgan Kaufmann, San Francisco (2006)
9. Skopal, T., Krátký, M., Snášel, V., Pokorný, J.: A New Range Query Algorithm for the Universal B-trees. Information Systems 31(6), 489–511 (2006)
10. Vitter, J.S.: Algorithms and Data Structures for External Memory. Series on Foundations and Trends in Theoretical Computer Science. Now Publisher (2008)

Construction of Messaging-Based Integration Solutions Using Constraint Programming

Pavol Mederly and Pavol Návrat

Faculty of Informatics and Information Technologies, Slovak University of Technology,
Ilkovičova 3, 842 16 Bratislava 4, Slovak Republic
{mederly,navrat}@fiit.stuba.sk

Abstract. This paper presents a novel method of designing selected aspects of messaging-based integration solutions. The method uses constraint programming to find appropriate communication channels, components' deployment parameters and integration services in order to create a solution that meets specified functional and non-functional requirements. The method has been evaluated using a prototype implementation and compared to authors' earlier work that used action-based planning techniques to reach similar goals.

Keywords: Enterprise Application Integration, Enterprise Integration Patterns, Messaging, Constraint Programming.

1 Introduction

Application integration is one of the crucial issues in enterprise computing [1]. Despite the existence of powerful tools, development of integration solutions (i.e. software systems that interconnect individual applications) is still an expensive task requiring highly-skilled software developers. Although there are some works trying to automate activities in the area of integration solutions' implementation (e.g. [5]) as well as abstract specification (e.g. [2]), we know of no attempts to automate designing such solutions.

This paper presents a novel method for automatic design of messaging-based integration solutions using constraint programming techniques. The method and its evaluation are described in sections 2 and 3, respectively. Section 4 closes this paper and gives some ideas on future work.

2 The Method for Automated Design of Integration Solutions

One of the important approaches to design of integration solutions is *messaging*. It allows individual applications to communicate by exchanging messages in an asynchronous and platform-independent way. Such solutions are often based on the Pipes and Filters architecture: they consist of processing components called filters that are connected using channels (pipes). Some of the filters correspond to applications that are being integrated, while others implement auxiliary services needed for the integration to take place. The former are sometimes called business services, while the latter integration (or mediation) ones.

The method's input is an *integration problem*: an abstract description of the integration solution that is to be implemented. It consists of information on business services that are to be incorporated into the solution, a logical flow of messages among these services, and a set of non-functional requirements to be met, namely:

– *throughput and availability* of the solution; as a part of the integration problem specification the throughput and reliability for each business service in each of four deployment modes (see below) are provided;
– a list of services whose inputs and outputs should be *monitored*;
– *message ordering* and *duplicate messages avoidance*, if it is necessary to guarantee these features at some points in the solution;
– *message formats* (e.g. XML, JSON, etc.) supported by individual services.

The method then tries to meet these requirements by creating a suitable design of the solution, namely by:

– *deploying components appropriately* – in the current version of the method we distinguish between deployment in (1) single thread, (2) multiple threads, (3) multiple processes and (4) multiple hosts;
– *choosing suitable channels* – we consider in-memory channels (cheap but limited in functionality) and two kinds of message broker-based ones: point-to-point (enabling distributed deployment of services) and publish-subscribe channels (enabling easy monitoring), optionally with transactional behaviour;
– *inserting appropriate integration services* – e.g. Wire Tap (for monitoring), Order Marker and Resequencer (for message ordering), Message Translator (for format conversion), Message Filter (for duplicate elimination) [1].

The method works by translating the integration problem into a constraint satisfaction problem (CSP) consisting of a set of variables and a set of constraints imposed upon them. An example of a structure of a result of the translation – for a part of the order processing scenario described in [3] – is shown in Fig. 1.

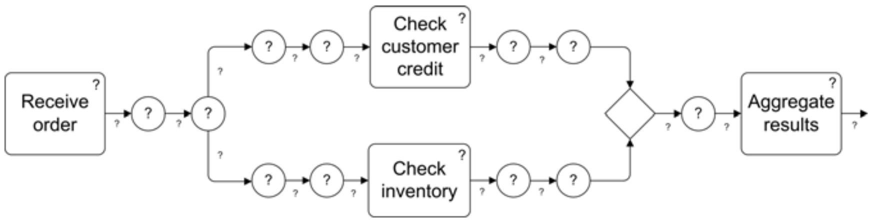


Fig. 1. An example of a structure of CSP corresponding to an integration problem

Rectangles represent business services. Circles represent slots for integration services that are to be found. The initial number of such slots is configurable; the method adjusts it while looking for the solution. Question marks attached to services and message flows point up to the following sets of CSP variables:

1. *For each integration or business service* there is a set of variables corresponding to: (1) the type of the service (only for integration services), (2) mode of deployment as described above, (3) whether this service takes part in a messaging infrastructure transaction, and (4) cost of deploying this service in a specified way.
2. *For each message flow*, i.e. a line connecting two services, there is a set of variables indicating: (1) type of channel, (2) whether the flow is in the original order, (3) whether messages have sequence numbers assigned, (4) whether duplicate messages can be possibly present, (5) whether the flow is monitored, and (6) the message format.

Constraints imposed upon CSP variables are used mainly to reflect requirements and effects of individual services on the respective message flows. They are created by applying predefined templates to all business and integration services. For example, constraints related to message ordering look like this:

```
IF service.Type != RESEQUENCER AND service.Type != NONE THEN
    (IF service.Deployment = SINGLE-THREAD THEN
         service.output.Ordered = service.input.Ordered) AND
    (IF service.Deployment != SINGLE-THREAD THEN
         service.output.Ordered = FALSE).
```

3 Implementation and Evaluation

The method has been implemented as a research prototype, utilizing the JaCoP solver. It has been evaluated using a set of integration problems taken from the literature as well as from our real-life experience (see Table 1).

The *Problem description* column references the respective integration problem as it is present in [4]. The *Aspects* column shows which of the design issues the problem deals with, namely: monitoring (M), message formats (F), throughput (T), availability (A), message ordering (O), and duplicate messages elimination (D). The *Business services* and *Int. comp.* columns show the number of business services and integration components used in the optimal solution, respectively. The last three columns provide the CPU time needed to find the optimal solution, to conclude that no better solution exists and to find a solution by our previous method, where applicable [3,4].

4 Conclusion and Future Work

Research results presented here demonstrate that it is possible to use constraint programming to automate the design of some aspects of integration solutions.

Table 1. Selected results of the evaluation of the method

#	Problem description	Aspects	Business services	Int. comp.	Optimal solution (sec)	All solutions (sec)	Planning (sec)
1	Reseller (D)	MFTA	5	10	0.17	0.44	0.43
2	Reseller (J)	MFTAO	11	20	0.33	30.04	15.47
3	Reseller (J)	MFTAOD	11	29	2.80	7,012.36	n/a
4	University (G)	MFTA	11	20	0.20	1.87	55.00

Our aims for the future include covering other design aspects, e.g. mapping logical data elements into messages and their parts, supporting diverse transport protocols or deploying solution components into an integration infrastructure. In order to achieve this goal we intend to combine AI planning and constraint programming approaches; we also plan to give the developer a possibility to influence the solution-finding process. We are working on a code-generation module as well.

Acknowledgments. This work was supported by the Scientific Grant Agency of SR, grant No. VG1/0508/09 and it is a partial result of the Research & Development Operational Program for the project Support of Center of Excellence for Smart Technologies, Systems and Services II, ITMS 25240120029, co-funded by ERDF.

References

1. Hohpe, G., Woolf, B.: Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Pearson Education, Inc., Boston (2004)
2. Mayer, W., Thiagarajan, R.K., Stumptner, M.: Service Composition as Generative Constraint Satisfaction. In: IEEE International Conference on Web Services, ICWS 2009, pp. 888–895. IEEE Computer Society, Los Alamitos (2009)
3. Mederly, P., Lekavý, M., Závodský, M., Návrat, P.: Construction of Messaging-Based Enterprise Integration Solutions Using AI Planning. In: Preprint of the Proceedings of the 4th IFIP TC2 Central and East European Conference on Software Engineering Techniques, CEE-SET 2009, pp. 37–50. AGH University of Science and Technology, Krakow (2009)
4. Mederly, P., Lekavý, M.: Report on Evaluation of the Method for Construction of Messaging-Based Enterprise Integration Solutions Using AI Planning, `http://www.fiit.stuba.sk/~mederly/evaluation.html`
5. Scheibler, T., Leymann, F.: A Framework for Executable Enterprise Application Integration Patterns. In: Mertins, K., et al. (eds.) Enterprise Interoperability III, pp. 485–497. Springer, London (2008)

Criteria for Reducibility of Moving Objects Closeness Problem

Elena Snegova[*]

Moscow State University
lenasnegova@gmail.com

Abstract. Consider two streams of moving objects inside the square $[0,1]^2$. We assume that objects in each of streams move in prescribed manner, but have random coordinate and random time of appearance. One of the streams moves from South to North and we call it a stream of queries, and another stream moves from West to East and we call it a stream of objects. Moving objects closeness problem (MOC problem) consists of enumeration for every new query those and only those objects that will be not far than ρ from the query in Manhattan metrics at some moment of time during the query's or the objects' movements inside the square.[1]

In general case this problem is very hard to solve because of dynamic situation and two-dimensional movements of objects and queries. But, in some cases the MOC problem is equivalent to one-dimensional range searching problem that can be solved effectively with logarithmic search, insertion and deletion time and a linear memory size as functions of the number of objects inside the square. In this paper we present and prove criteria for reducibility of the MOC problem to one-dimensional range searching problem.

1 Introduction

Traditional database management systems assume that every moving object has its own independent velocity and direction of movement, i.e. objects move chaotically. Our approach is to consider streams of objects and to assume conflicts between streams, but not inside one stream.

In this paper we consider the case of prescribed object movement. The close case was also considered in [1] by Attalah. The author proposed algorithms for searching the closest and fastest points and for computing convex hull of points.

The information about the laws of motion of objects and queries let us define in every specific case, if the MOC problem can be reduced to one-dimensional range searching problem or not. In the next chapter we present criteria for reducibility.

[*] The author warmly thanks his supervisor, Prof. E.E. Gasanov, for problem statement and invaluable help.

[1] The distance in Manhattan Metrics can be interpreted in the following way: consider moving squares (for example, airplanes) with "radius" $\frac{\rho}{2}$ as it is shown on Fig. 1. In MOC problem for every moving squared query it is required to list all squared objects that will overlap the query during its movement inside the square.

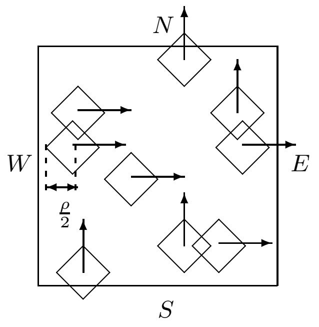


Fig. 1. The movement of objects and queries in MOC problem

There are a lot of storage methods appropriate for one-dimensional range searching, for example [3,4,5,6,7,8]. For dynamic case the most suitable is the method based on 2–3 tree structure [2]. The analytical estimates of corresponding algorithm for searching, insertion and deletion time are much lower in compare with Attalah's estimates [1].

2 Basic Notions and Formulation of the Result

Let τ_{max} be object movement time inside the square $[-\rho, 1+\rho]\times[-\rho, 1+\rho]$ and let τ^q_{max} be query movement time inside the same square. Let $f : [0, \tau_{max}] \to [-\rho, 1+\rho]$ be object law of motion and let $f_q : [0, \tau^q_{max}] \to [-\rho, 1+\rho]$ be query law of motion. (We consider the movement of objects and queries in a broader rectangle than $[0, 1]^2$, because in an answer to every query it is required to enumerate all objects objects that will be not far than ρ from the query in Manhattan metrics at some moment of time during the query's or the objects' movements inside the square.) Assume f and f_q are continuous increasing functions such that $f(0) = f_q(0) = -\rho$, $f(\tau_{max}) = f_q(\tau^q_{max}) = 1 + \rho$.

Suppose there is a countable set of objects on the plane. Then, every object inside $[-\rho, 1 + \rho] \times [-\rho, 1 + \rho]$ at time t is a pair $(f(t - t_i), y_i)$, where $i \in N$, $y_i \in [0, 1]$ is a coordinate of appearance on axis Y of object i, and the moments of appearance t_i form an increasing sequence of positive numbers. Similarly, every moving query at time t is a pair $(x, f_q(t - t_q))$, where $x \in [0, 1]$ is a query's coordinate of appearance on axis X and $t_q \geq 0$ is a query's moment of appearance. The position of every object and every query depends on the moment of appearance, coordinate of appearance and corresponding law of motion. Then let us consider objects as pairs $o_i = (t_i, y_i)$ and queries as pairs $q = (t_q, x)$.

Library $V(t_q)$ is a set of all objects inside $[-\rho, 1 + \rho] \times [0, 1]$ at the current time t_q, i.e. $V(t_q) = \{(t_i, y_i) : t_q \in [t_i, t_i + \tau_{max}], y_i \in [0, 1]\}$. Sometimes, instead of $V(t_q)$ we will use the notation V that implies the set of all objects inside $[-\rho, 1 + \rho] \times [0, 1]$ at the current time. Let $|V|$ be the number of objects in the current library.

Given the library $V = V(t_q)$, the answer for the query $q = (t_q, x)$ is the set of objects from V, that will be in closeness with the query inside $[-\rho, 1+\rho] \times [-\rho, 1+\rho]$, i.e. $J(\rho, q, V) = \{o_i \in V | \ \forall \ i \ \exists \ t : |f(t-t_i)-x|+|y_i-f_q(t-t_q)| \le \rho\}$.

The triple (f, f_q, ρ) is called *the moving objects closeness problem (the MOC problem).*

One-dimensional range searching problem is a pair (I, Z), where library Z is a finite subset of the set of real numbers R and a query set I is a set of segments from R. Meaningfully one-dimensional range searching problem consists in finding for any segment p from I those and only those points from Z that belongs to the segment p, i.e. the answer for the query $p \in I$ is the set $J(p, Z) = \{z \in Z : z \in p\}$.

We say that the MOC problem (f, f_q, ρ) *can be reduced* to one-dimensional range searching problem iff there exists functions $\varphi, \varphi_1, \varphi_2 : R \times [0,1] \to R$ such that for any library $V = V(t_q)$, any query $q = (t_q, x)$ and any object $o \in V$ the following is true $o \in J(\rho, q, V) \Leftrightarrow \varphi(o) \in J([\varphi_1(q), \varphi_2(q)], Z)$, where $Z = \{\varphi(o) : o \in V\}$.

Denote:

$$F_L(x,y) = \min_{\xi \in [0,\rho]} [f_q^{-1}(y+\xi-\rho) - f^{-1}(x+\xi)],$$

$$F_R(x,y) = \max_{\xi \in [-\rho,0]} [f_q^{-1}(y+\xi+\rho) - f^{-1}(x+\xi)].$$

Theorem 1. *The moving objects closeness problem (f, f_q, ρ) can be reduced to one dimensional range searching problem iff there exist functions $\psi, \psi_L, \psi_R : [0,1] \to R$ such that*

$$F_L(x,y) = \psi(y) + \psi_L(x) \ \ \forall (x,y) : F_L(x,y) \le 0, \tag{1}$$

$$F_R(x,y) = \psi(y) + \psi_R(x) \ \ \forall (x,y) : F_R(x,y) \le 0. \tag{2}$$

Four characteristics (the size of memory M, search time (without the answer enumeration time) T, insertion time S and deletion time D) can be used for the estimation of information processing algorithms. We measure time as a number of elementary operations made during the correspondent procedure.

Functional complexity is a function of dependence of these characteristics upon the number of objects in the library.

We say that $h(n) = O(g(n))$, if there exists constants $C_1 > 0, C_2 > 0$, and constants C_3, C_4, such that for all n the following condition is satisfied:

$$C_1 g(n) + C_3 \le h(n) \le C_2 g(n) + C_4.$$

Corollary 1. *Assume that for the triple (f, f_q, ρ) there exist functions $\psi, \psi_L, \psi_R : [0,1] \to R$ such that conditions (1)and (2) are satisfied. Then there exists algorithm A that solves the MOC problem (f, f_q, ρ). For algorithm A complexities of search (without answer enumeration time), insertion, deletions procedures and for size of memory of algorithm A the following estimates are valid correspondingly:*

$$T_A(V) = O(\log_2 |V|),$$

$$S_A(V) = O(\log_2 |V|),$$
$$D_A(V) = O(\log_2 |V|),$$

$$M_A(V) = O(|V|),$$

where operations of addition, comparison of two numbers and operation of computation ψ, ψ_L, ψ_R of every point from $[0,1]$ are taken as elementary.

3 Example

Consider the case, when $f'(\tau) \leq f'_q(\tau')$ for all $\tau \in [0, \tau_{max}]$ and $\tau' \in [o, \tau^q_{max}]$. Then, $F_L(x,y) = f_q^{-1}(y) - f^{-1}(x+\rho)$ and $F_L(x,y) = f_q^{-1}(y) - f^{-1}(x-\rho)$. According to the criteria it means that the MOC problem (f, f_q, ρ) with this property can be reduced to one-dimensional range searching problem. It is possible to show that according to the definition of the answer for the query an object $o_i = (t_i, y_i)$ from $V = V(t_q)$ belongs to $J(\rho, q, V)$ where $q = (t_q, x)$ iff $t_i - t_q \in [F_L(x, y_i), F_R(x, y_i)] \Leftrightarrow t_i - f_q^{-1}(y) \in [t_q - f^{-1}(x+\rho), t_q - f^{-1}(x+\rho)]$. Thus, for one dimensional range searching problem $[t_q - f^{-1}(x+\rho), t_q - f^{-1}(x+\rho)]$ is a query and $t_i - f_q^{-1}(y)$ is an element of the library at the moment t_q.

References

1. Atallah, M.: Dynamic Computational Geometry. In: Proc. 24th Annu. IEEE Sympos. Found. Comput. Sci. (1983)
2. Lapshov, I.S.: Dynamic databases with optimal in order time complexity. Discrete Mathematics and Applications 18(4), 367–379 (2006)
3. Bentley, J.L., Friedman, J.H.: Data structure for range searching. Comput. Surveys 11, 397–409 (1979)
4. Bolour, A.: Optimal Retrieval Algorithms for Small Regional Queries. SIAM J. Comput. 10, 721–741 (1981)
5. Bernard, C.: Filtering Search: A new approach to query-answering. SIAM J. Comput (1986)
6. Fredman, M.L.: A lower bound of complexity of ortogonal range queries. J. ACM 28, 696–705 (1981)
7. Leuker, G.S.: A data structure for ortogonal range queries. In: Proceedings of 19th Annual IEEE Sympothium on Foundations of Computer Science, pp. 28–34 (1978)
8. Willard, D.E.: Predicate-oriented database search algorithms. Ph.D. dissertation, Harvard Univ., Cambridge, MA (1978)

Platform Independent Database Replication Solution Applied to Medical Information System

Tatjana Stankovic[1], Srebrenko Pesic[2], Dragan Jankovic[1], and Petar Rajkovic[1]

[1] Faculty of Electronic Engineering, Aleksandra Medvedeva 14, 18000 Nis, Serbia
{tatjana.stankovic,dragan.jankovic,
petar.rajkovic}@elfak.ni.ac.rs
[2] Health Centre Nis, Vojvode Tankosica 15, 18000 Nis, Serbia
srebrenko.pesic@domzdravljanis.co.rs

Abstract. Medical Information System (MIS), as any other information system (IS), can be based on centralized or distributed database management system. It is much easier to implement and to administer centralized system, but the benefits of distributed over centralized architecture in our case were crucial. However, database replication in distributed MIS represents a significant problem. All well-known database replications are platform-dependent, and when it comes to the possible platform migration, a huge problem of setting replication on different platforms is born. This paper describes one platform independent - information system dependent database replication solution, that can enable fast replication even in low-band/low-speed internet connections.

Keywords: Database replication, replication conflicts, distributed DBMS.

1 Introduction

Modern society implies the existence of highly developed public health system with a strong technical and organizational background. Main building blocks of mentioned technical background are effective and reliable Medical Information Systems[1] in active use within healthcare facilities having high level of interoperability [1].

As any other complex and large IS, MIS can be either based on centralized or distributed database management system (DDBMS). According to the rules that Serbian Ministry of Health defined [2], in our country both are allowed, as soon as the fact that "the system must be available 24 hours a day, 365 days a year" is satisfied. Centralized system is more vulnerable then distributed in a sense of connectivity. Larger health centres in Serbia are usually consisted of one central node and several tens of remote nodes (small clinics). There are local clinics in schools, villages, etc. Network connections between those clinics, in many cases, are not reliable; it is pretty much possible to have some clinic disconnected for hours, even days.

DDBMS solves mentioned problems, but it demands very well planned database replication. Setting and administering database replication on so many DBMSs costs

[1] Research and MIS described in this paper are supported by the Ministry of Science and technological Development of the Republic of Serbia (Project Nr TR13015).

many expert-hours. Resolving database replication conflicts demands good planning, and even then, practice can lead us to unresolved replication conflicts that end in a dead queue, waiting for a human interaction. Besides, there is a standard proposed by Republic of Serbia as follows: any MIS solution must ensure that technological platform for the solution is independent on DBMS platform [2]. Known reliable replication solutions such are MS SQL Server Replication, Oracle replication, Slony-I, are all platform-dependant. For reasons mentioned above we have developed one platform independent – IS dependant database replication solution.

2 Information System Dependant Database Replication

The main idea in designing database platform independent replication scenarios and systems was to plan replication and solve replication conflicts in advance, in the MIS planning phase.

Our main intense was to keep all replication required data inside of DBMSs, but not in a way as existing solutions usually do. For example, SQL Server adds new unique field in every table involved in transactional replication from the publisher's side [3]. In our case, there are only 5 tables in every database that are directly responsible for replication.

MIS deployed in a single healthcare facility can produce millions of records per table yearly [4]. Fortunately, not every single DBMS needs to be balanced with all that data, and in our replication, that balance is determined in advance by MIS.

The first priority was modelling dependences between DDBMS architecture and healthcare facility organization structure, as is shown in Figure 1. Both structures are presented as n-ary trees. The root of DDBMS n-ary tree is central server, which usually resides in central unit of healthcare facility.

Primary key replication conflicts are solved in advance by implementing offsets for primary keys in every database [5]. UPDATE/DELETE replication conflicts are impossible, because MIS takes good care of data ownership [5].

Basic concept can be briefly described in following way: MIS logs every INSERT, UPDATE or DELETE statement executed against database, together with some additional parameters (what is destination DBMS in n-ary tree for that specific information, is information to be sent to all servers, etc). All loggings are done in the same database. Segment of our database, responsible for replication, is shown in Figure 1. When it comes to data synchronization, statements logged after the moment of last performed synchronization are selected, packed in a text file, and sent to the other replication side (simplified explanation). The statements are ordered by timestamp of execution against primary database. Of course, not all statements are selected, but only those that correspond to the predefined rules (table is or is not involved in replication between two servers is only one of many rules). On another replication side, the file is unpacked, and the statements are executed against that database, inside of one large transaction. According to predefined algorithm applied to every statement, the statements are logged (or not logged) for further replication. The moment when statements will be archived and removed from log table, depends on replications performed earlier, and timestamp that corresponds to the statement.

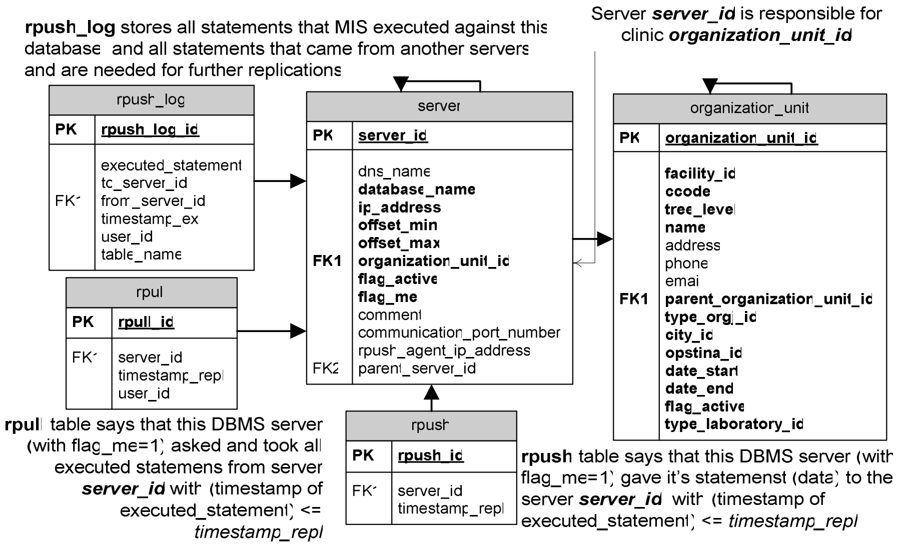

rpush_log stores all statements that MIS executed against this database and all statements that came from another servers and are needed for further replications
Server server_id is responsible for clinic organization_unit_id
rpush_log
PK rpush_log_id
FK1 executed_statement
to_server_id
from_server_id
timestamp_ex
user_id
table_name
server
PK server_id
dns_name
database_name
ip_address
offset_min
offset_max
FK1 organization_unit_id
flag_active
flag_me
comment
communication_port_number
rpush_agent_ip_address
FK2 parent_server_id
organization_unit
PK organization_unit_id
facility_id
ccode
tree_level
name
address
phone
email
FK1 parent_organization_unit_id
type_org_id
city_id
opstina_id
date_start
date_end
flag_active
type_laboratory_id
rpull
PK rpull_id
FK1 server_id
timestamp_repl
user_id
rpush
PK rpush_id
FK1 server_id
timestamp_repl
rpull table says that this DBMS server (with flag_me=1) asked and took all executed statemens from server server_id with (timestamp of executed_statement) <= timestamp_repl
rpush table says that this DBMS server (with flag_me=1) gave it's statemenst (data) to the server server_id with (timestamp of executed_statement) <= timestamp_repl

Fig. 1. 5 tables implemented in every DBMS, responsible for replication

3 Replication Agents

Replication procedure is done by two replication agents (client, called Rpull agent, and server, called Rpush agent). They communicate over TCP/IP on a port that is defined in advance between administrators [6], and log information about successfully performed replications. Server agent can serve more than one client agent simultaneously (multithreading). The process is shown in Figure 2.

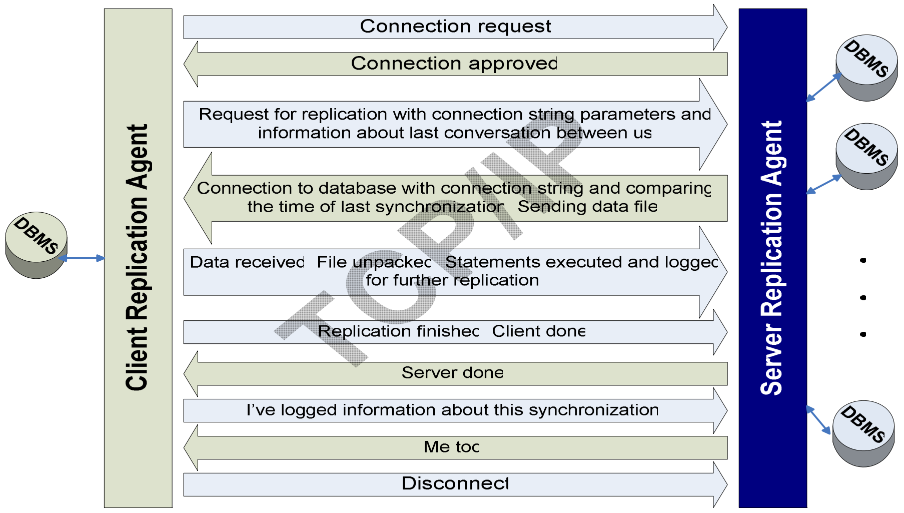

Client Replication Agent
Server Replication Agent
DBMS
Connection request
Connection approved
Request for replication with connection string parameters and information about last conversation between us
Connection to database with connection string and comparing the time of last synchronization. Sending data file
Data received. File unpacked. Statements executed and logged for further replication
Replication finished. Client done
Server done
I've logged information about this synchronization
Me too
Disconnect
TCP/IP

Fig. 2. Replication process workflow

4 Conclusions and Remarks

MS SQL Server 2005 was database platform we have started to work with. MS SQL Server Standard Edition supports all types of replication (snapshot, transactional, merge, merge bidirectional), but MS SQL Server Express, which is non-commercial and convenient for smaller remote clinics, has very limited scope of database replication [7]. Having a solution like presented in this paper may represent a progress to our project. There is a possibility to use the combination of platform dependant replication - for objects like medical catalogues, that are replicated entirely over the whole DBMS n-ary tree, and our platform-independent simple replication for those whose setting is complicated (like shipments of medical materials and drugs, patient examination data, etc). The other item is related to data whose existence is not necessary in all tree nodes, but only to specific ones, determined in advance. This means much less data waste for wayside servers, and reliable replication in low-band/low-speed network connections.

And at the end, this kind of replication can easily be applied to any IS beside MIS we have developed, by taking several actions. First: 5 replication tables need to be added to the database. Organization structure of the company must be designed as an n-ary tree (which is the case in most companies). Data layer need to be modified by implementing a bottom-level function that will determine logging of executed statements. And finally, replication agents need to be set to the nodes for replication.

References

1. Wager, K.A., Wickham Lee, F., Glaser, J.P.: Managing Healthcare Information Systems - A Practical Approach for Health Care Executives, 1st edn., May 5. Jossey-Bass, San Francisco (2005), ISBN-10:078797468
2. Article 3, paragraph 3, Uredba o Programu rada, razvoja i organizacija integrisanog zdravstvenog informacionog sistema "e-Health", Sluzbeni glasnik RS, br. 55/09
3. Sujou, P.: Pro SQL Server 2005 Replication, Springer-Verlag New York, ISBN-13: 978-1-59059-650-0, ISBN-10: 1-59059-650-1, United States of America 987654321
4. Pesic, N.S., Stankovic, N.T., Jankovic, S.D.: Benefits of Using OLAP Versus RDBMS for Data Analyses in Health Care Information Systems. Electronics 13(2), 56–60 (2009)
5. Stankovic, N.T., Jankovic, S.D., Pesic, N.S.: Public Health Care Distributed DBMS with Resolving Database Replication Conflicts in the Health Care Information System Project Early Phase. In: 9th International Conference on Telecomunications in Modern Satellite, Cable and Broadcasting Services - TELSIKS, IEEE Catalogue Number: CFP09488-PRT, Nis, Serbia, vol. 2, pp. 487–490 (2009)
6. Stankovic, N.T., Stankovic, R.M., Jankovic, S.D.: Platform Independent-Information System Dependent Database Replication. In: 43th International Scientific Conference On Information, Communication and Energy Systems and Technologies - ICEST, Nis, pp. 415–418 (2008)
7. http://msdn.microsoft.com/en-us/library/ms165686.aspx

Indexing Temporal Data with Virtual Structure

Bela Stantic, Justin Terry, Rodney Topor, and Abdul Sattar

Institute for Integrated and Intelligent Systems
Griffith University, Brisbane, Australia
{B.Stantic,J.Terry,R.Topor,A.Sattar}@griffith.edu.au

Abstract. Temporal and spatio-temporal data are present in many modern application systems, including monitoring moving objects. Such systems produce enormous volume of data, and therefore efficient indexing method is crucial. In this paper, we investigate and present a new concept based on virtual index structure, which can efficiently query such data. Concept is based on spatial representation of interval data and a recursive triangular decomposition of that space. The empirical performance of presented concept is demonstrated to be superior to its best known competitors.

1 Introduction

Modern database system, such as used for tracking moving objects, contain a significant amount of time dependent data [6]. Over the last two decades due to increased number of spatio-temporal applications interest in the field of temporal databases has increased significantly with contributions from many researchers [1], [8]. Because temporal databases are in general append only, they are usually very large in size, thus efficient access method is even more important in temporal databases than in conventional databases [2]. A number of access methods for temporal data that utilise the relational database systems built-in functionalities have been proposed, however, they have either space complexity problem or are generally tailored to be efficient only for specific query types. In [5] it has been shown that the RI-tree is superior to main competitors, the Window-List, Oracle Tile Index (T-Index) and IST-technique. However, the RI-tree need to tailor query transformation to the specific query types. It is our intention to propose an efficient access method for temporal data with logarithmic access time and guaranteed minimum space complexity that can answer a wide range of query types with the same query algorithm.

In this paper we present and investigate the Triangular Decomposition Tree (TD-tree) access method to index and query temporal data. In contrast to previously proposed access methods for temporal data this method can efficiently answer a wide range of query types, including point queries, intersection queries, and all nontrivial interval relationships queries, using a single algorithm without dedicated query transformations. It also can be built within commercial relational database system as it uses only built-in functionalities within the SQL:1999 standard and therefore no modification to the database kernel is required.

The TD-tree is a space partitioning access method. The basic idea is to manage the temporal intervals by a virtual index structure that relies on a two-dimensional representation of intervals and a triangular decomposition method. The resulting virtual binary tree stores a bounded number of intervals at each leaf and hence may be unbalanced. As data is only stored in leaves, traversing the tree avoids disk accesses and tree depth therefore does not affect performance. Using the interval representation, any query type can be reduced to a spatial problem of finding those (triangular) leaves that intersect with the spatial query region. The efficiency of the TD-tree is due to the virtual internal structure so there is no need for physical disk I/O's, query algorithm that ensures pruning, and efficient clustering of interval data. On top of the advantages related to the usage of a single query algorithm for different query types and better space complexity the empirical performance of the TD-tree is demonstrated to be superior to its best known competitors.

2 The Triangular Decomposition Tree (TD-Tree)

The structure of our indexing method is based on the observation, that all data and query intervals of interest represented in two dimensional space lie in the isosceles, right-angle triangle with vertices at (0,0), $(0,\lambda)$ and (λ,λ), which lies above the line $E = S$. We call this triangle the *basic triangle* Figure 1. This is due to nature of interval space transformation and fact that $i_s < i_e$.

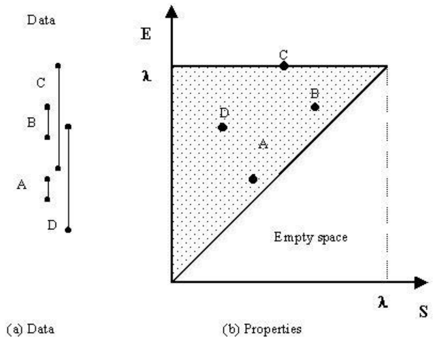


Fig. 1. Interval representation in two-dimensional space

Given that our region of interest is a triangle, our proposal is to recursively decompose the basic triangle into two smaller triangles, whenever triangle covers more than the chosen number of interval objects defined by blocking factor b. In such a triangular decomposition, each triangle is uniquely identified by its *apex* position (s,e), and its *direction* d, the direction of the arrow from the midpoint of the triangle's hypotenuse to the apex.

Given a triangle in this decomposition, its apex and direction uniquely determine the apex and direction of each of its two subtriangles. The position (s,e) of the apex of each subtriangle C of a parent triangle P at any level l is possible to find out by knowing only the position of the parent apex and its level. Because we can identify the apex and direction of every node of a TD-tree, starting from basic triangle *we do not need to store the internal tree nodes*. Thus, a TD-tree is a virtual tree. If a node has identifier ϕ, the lower and upper children of the node have identifiers $\phi 0$ and $\phi 1$ respectively. The length of the identifier is thus one greater than the depth of the node. Information about leaf nodes themselves are stored in a separate directory, containing an identifier and number of records per leaf. The root node stores the blocking factor $\mathtt{b}$, λ, and current maximum depth of the tree.

It can be shown that the every query corresponds to a rectangular region of the two-dimensional interval space, defined by the top-left and bottom-right corners of this region. The task of query evaluation is to find all data intervals that occur within this query region. The particular region chosen depends on whether we are performing an intersection query, an overlaps query, a contains query, and so on, but in each case the query evaluation algorithm is identical, this is an important property of our approach.

3 Experimental Evaluation

To show the practical relevance of our approach, we performed an extensive experimental evaluation of the TD-tree and compared it to the RI-tree [5]. The RI-tree was chosen, since it provides the same practically important properties as our approach. It is easy to implement and integrate, it uses standard RDBMS methods which provides scalability, update-ability, concurrency control and space efficiency. Furthermore it has been shown [5] that the RI-tree is superior to main competitors so the performance results of the TD-tree can be transferred to these indexing techniques. In our experiment we tested performance on intersection queries. However, because of the nature of our query algorithm, which compares the data region with the rectangular query region, results for intersection query applies to the other query types.

The Theory of Indexability [3] identifies I/O complexity cost, measured by the *number of disk accesses*, as one of the most important factors for measuring query performance, which in conjunction with *CPU usage* is used to assess the performance of the query process.

When making performance measurements of index structures it is important to consider parameters such as space requirements, physical disk I/O, CPU usage, clustering, updates, and locking. In our analysis we have concentrated on space requirements, physical disk reads, CPU usage and clustering of data. Because both the RI-tree and TD-tree rely on the relational paradigm, updates and locking are handled well by the RDBMS itself.

The TD-tree requires only one virtual index structure, which means only leaf nodes have to be stored. The list of leaf nodes are stored in the directory table and its size is very small comparing to the table itself. In our experiment the TD-tree directory for one million interval objects required only 26 data blocks. While The RI-tree requires two composite index structures *lowerIndex* (on *node* and *Start* - start of the interval) and the *upperIndex* (on *node* and *End* - end of the interval). In our experiment the RI-tree composite indexes for one million interval objects required 6708 data blocks (3354 each index).

The TD-tree enables efficient usage of clustering of the data by one dimension, i.e region, as every region associate with block size. Clustering data improves the query performance and reduces the number of physical I/O, clustering ensures higher number of answers per physical disk I/O. In contrast, the RI-tree can not efficiently use clustering of data as it has to decide which dimension to use start or end. If it is clustered by *node* it will not result in similar improvements, as in RI-tree *node* are fixed size and are too large to provide effective clustering.

4 Conclusions

We described a new approach to efficiently manage vast volume of temporal and spatio-temporal data within the commercial RDBMS. More specifically, in this paper we:

- Presented the triangular decomposition tree (TD-tree) concept that can efficiently answer a wide range of query types with single algorithm;
- Experimentally evaluated the TD-tree by comparing its performance with the RI-tree, and demonstrated its overall superior performance.

The TD-tree is a unique access method as it uses tree structure and at the same time has some characteristics of hashing because it only stores data in leaf nodes. As a wide range of query types of interest can be reduced to rectangular region, it is possible to answer such queries using a single algorithm without requiring any query transformation. This itself, and the fact that the TD-tree can be incorporated within commercial RDBMS and utilised in a lot of spatio-temporal applications that manage vast volume of data, makes the TD-tree superior to other methods currently used or proposed in literature. Additionally, presented concept of regular decomposition and virtual internal structure can be extended and applied to more dimensions to efficiently manage high dimensional data.

References

1. Date, C., Darwen, H., Lorentzos, N.: Temporal Data and the Relational Model. Morgan Kaufmann, San Francisco (2002)
2. Dyreson, C.E., Snodgrass, R.T., Freiman, M.: Efficiently Supporting Temporal Granularities in a DBMS. Technical Report TR 95/07 (1995)
3. Hellerstein, J., Koutsupias, E., Papadimitriou, C.: On the Analysis of Indexing Schemes. In: 16th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (1997)
4. Kriegel, H.-P., Potke, M., Seidl, T.: Object-relational indexing for general interval relationships. In: Proc. 7th Int'l Symposium on Spatial and Temporal Databases, SSTD 2001 (2001)
5. Kriegel, H.-P., Potke, M., Seidl, T.: Managing intervals efficiently in object-relational databases. In: Proceedings of the 26th International Conference on Very Large Databases, pp. 407–418 (2000)
6. Marios, H., Kollios, G., Tsotras, V.J., Gunopulos, D.: Indexing Spatialtemporal Archives. The VLDB Journal 15(2) (2006)
7. Ramaswamy, S.: Efficient Indexing for Constraint and Temporal Databases. In: Proceedings of the 6th International Conference on Database Theory, pp. 419–431 (1997)
8. Snodgrass, R.T.: Developing Time-Oriented Database Applications in SQL. Morgan Kaufmann, San Francisco (2000)

Detecting XML Functional Dependencies through Formal Concept Analysis

Katalin Tunde Janosi-Rancz[1], Viorica Varga[2], and Timea Nagy[2]

[1] Hungarian University of Transylvania, Tirgu Mures, Romania
tsuto@ms.sapientia.ro
[2] Babeş-Bolyai University, Cluj-Napoca, Romania
ivarga@cs.ubbcluj.ro

Abstract. We propose a framework which analyses an XML document constructing the Formal Context of the functional dependencies for the XML data. The obtained Concept Lattice is a useful graphical representation of the analyzed XML document's elements and their hierarchy. The software also finds functional dependencies and keys in XML data.

1 Introduction

Formal Concept Analysis (FCA) [2] is a method for data analysis, knowledge representation and information management that was successfully used in many fields, in relational database theory too. Our work is based on the definitions of the Generalized Tree Tuple, XML functional dependency (XFD) and XML key notion presented by [6]. This paper proposes a framework to mine functional dependencies from an XML database by reformulating the XML functional dependency inference with an FCA viewpoint. We construct the formal context of the functional dependencies for a tuple class or for the whole XML document. Non-leaf and leaf level elements (or attributes) and corresponding values are inserted in the formal context, then the concept lattice for this context is built up. The obtained Conceptual Lattice is the basis for further analysis. The set of implications resulted from this concept lattice will be equivalent to the set of functional dependencies that hold in the XML database.[1]

2 Overview of the Approach

Our approach is carried out by a sequence of processing steps and it is supported by a framework named FCAMineXFD. The examples of this paper are based on XML tree tuple of Fig. 1.

2.1 Constructing the Formal Context, the Input to FCA

As a first step, we need to define the objects and attributes of interest and create models of XML in terms of FCA context. Our software can analyze the whole

[1] Partially supported by Grant CNRS Nr. 55/2008

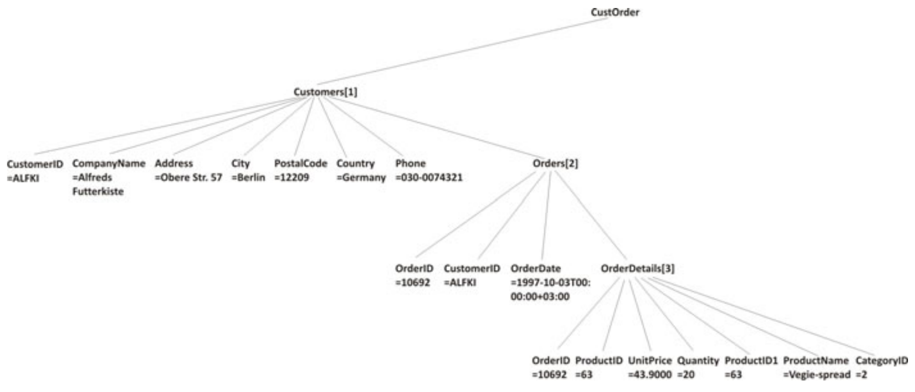


Fig. 1. Example tree tuple

XML document or a *tuple class* C_p given by the path p (see [6] for notations). Tuple-based XML FD notion proposed in [6] suggests a natural technique for XFD discovery. XML data can be converted into a fully unnested relation, a single relational table, and apply existing FD discovery algorithms directly. Given an XML document, which contains at the beginning the schema of the data, we create generalized tree tuples from it. Each tree tuple in a tuple class has the same structure, so it has the same number of elements. We use the flat representation which converts the generalized tree tuples into a flat table. Each row in the table corresponds to a tree tuple in the XML tree. In the flat table there are non-leaf and leaf level elements (or attributes) introduced from the tree. For non-leaf level nodes the associated keys are used as values.

Example 1. We construct the flat table for tuple class C_{Orders}. There are two non-leaf nodes: Orders and OrderDetails. These appear as attributes Orders@key and OrderDetails@key in the flat table.

Applying our experience in detecting functional dependencies in relational databases (see [4]), we use the definitions introduced by Hereth in [3]. Hereth gives the translation from the relational database into a power context family and based on it he defines the formal context of functional dependencies.

In this step the formal context of functional dependencies for XML data is built, mapping from metamodel entities to FCA objects and attributes.

Choice of FCA Attributes: *PathEnd/ElementName.* Due to space considerations we will not specify the whole path to the element (or attribute) names, only the end of the path. FCA attribute names are built from the end of the path to the element: *PathEnd* and element name as follows: for non-leaf level nodes the name of the attribute is constructed as: <ElementName>+"@key" and its value will be the associated key value. In case of the leaves the element names of the leaves of the tree tuple are considered.

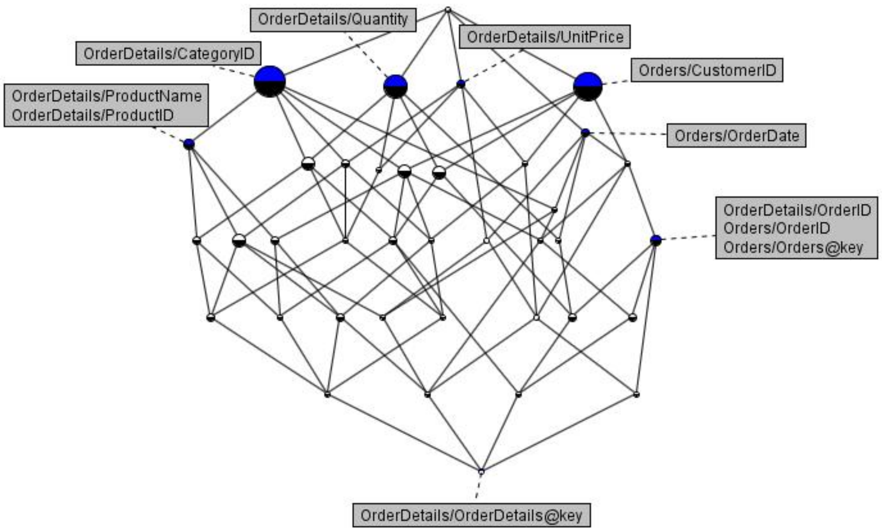


Fig. 2. Concept Lattice of functional dependencies' Formal Context for tuple class C_{Orders}

Choice of Objects: the objects are considered to be the tree tuple pairs, actually the tuple pairs of the flat table. The key values associated to non-leaf elements and leaf element's values are used in these tuple pairs.

Choice of Properties: the mapping between objects and attributes is defined by a binary relation, this incidence relation of the context shows which attributes of this tuple pairs have the same value.

2.2 Creating the Concept Lattice

Once the objects and attributes of the context are defined, we run the Concept Explorer ([5]) engine to generate the concepts and create the concept lattice. For tuple class C_{Orders} the corresponding concept lattice is shown in Fig. 2.

2.3 Processing the Output of FCA

A concept lattice consists of the set of concepts of a formal context and the subconcept-superconcept relation between the concepts, see [2]. Every circle in Fig. 2 represents a formal concept. An edge connects two concepts if one implies the other directly. For example node labeled with $Orders/CustomerID$ is on upward path from node labeled by $OrderDetails/OrderID$, $Orders/OrderID$, $Orders/Orders@key$, so concept with label $OrderDetails/OrderID$, $Orders/$ $OrderID$, $Orders/Orders@key$ implies concept with label $Orders/CustomerID$.

2.4 Mining XFDs According to the Concept Hierarchy

In this step, we examine the candidate concepts resulting from the previous steps and use them to explore XFDs. Once the lattice is constructed, we can interpret each concept and generate the list of all functional dependencies.

The relationship between FDs in databases and implications in FCA was pointed out in [2]: a FD $X \rightarrow Y$ holds in a relation r over R iff the implication $X \rightarrow Y$ holds in the context (G, R, I) where $G = \{(t_1, t_2) | t_1, t_2 \in r, t_1 \neq t_2\}$ and $\forall A \in R$, $(t_1, t_2)IA \Leftrightarrow t_1[A] = t_2[A]$.

This means that objects of the context are couples of tuples and each object intent is the agree set of this couple. Thus, the implications in this lattice corresponds to functional dependencies in XML. Therefore, we say that FCA can serve as a guideline for dependency mining.

Analyzing the Conceptual Lattice obtained for tuple class C_{Orders} (Fig. 2) FCAMineXFD detects functional dependencies like:

$\langle C_{Orders}, ./OrderID, ./CustomerID\rangle$, $\langle C_{Orders}, ./Orders@key, ./CustomerID\rangle$
$\langle C_{Orders}, ./OrderDetails/ProductID, ./OrderDetails/ProductName\rangle$,
$\langle C_{Orders}, ./OrderDetails/ProductID, ./OrderDetails/CategoryID\rangle$

2.5 Finding XML Keys

The implications found by FCAMineXFD contain some FDs with RHS as ./@key values. These can be used to detect the keys in XML. In tuple class C_{Orders} we have XML FD: $\langle C_{Orders}, ./OrderID, ./@key\rangle$, which implies that $\langle C_{Orders}, ./OrderID\rangle$ is an XML key. Another XML FD is $\langle C_{Orders}, ./OrderDetails/OrderID, ./@key\rangle$, so $\langle C_{Orders}, ./OrderDetails/OrderID\rangle$ is an XML key too.

References

1. Arenas, M., Libkin, L.: A normal form for XML documents. TODS 29(1), 195–232 (2004)
2. Ganter, B., Wille, R.: Formal Concept Analysis. In: Kutyłowski, M., Wierzbicki, T., Pacholski, L. (eds.) MFCS 1999. LNCS, vol. 1672, Springer, Heidelberg (1999)
3. Hereth, J.: Relational Scaling and Databases. In: Priss, U., Corbett, D.R., Angelova, G. (eds.) ICCS 2002. LNCS (LNAI), vol. 2393, pp. 62–76. Springer, Heidelberg (2002)
4. Janosi Rancz, K.T., Varga, V., Puskas, J.: A Software Tool for Data Analysis Based on Formal Concept Analysis. Studia Babeş-Bolyai, Informatica 53(2), 67–78 (2008)
5. Yevtushenko, S.A.: System of data analysis "Concept Explorer". In: Proceedings of the 7th National Conference on Artificial Intelligence KII-2000, Russia, pp. 127–134 (2000) (in Russian)
6. Yu, C., Jagadish, H.V.: XML schema refinement through redundancy detection and normalization. VLDB J. 17(2), 203–223 (2008)

Author Index